

教育訓練課程

Map Reduce

© TemplatesWan.com

王耀聰 陳威宇

Jazz@nchc.org.tw

waue@nchc.org.tw



財團法人國家實驗研究院

國家高速網路與計算中心

NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING

Computing with big datasets

is a fundamentally different
challenge than doing “big
compute” over a small dataset

平行分散式運算

- Grid computing
 - ◆ MPI, PVM, Condor...
- 著重於: 分散工作量
- 目前的問題在於: 如何分散資料量
 - ◆ Reading 100 GB off a single filer would leave nodes starved – just store data locally

分散大量資料：Slow and Tricky

- 交換資料需同步處理
 - ◆ Deadlock becomes a problem
- 有限的頻寬
 - ◆ Failovers can cause cascading failure

數字會說話

- Data processed by Google every month: 400 PB ... in 2007
 - ◆ Max data in memory: 32 GB
 - ◆ Max data per computer: 12 TB
 - ◆ Average job size: 180 GB
- 光一個device的讀取時間= 45 minutes

所以 ...

- 運算資料可以很快速，但瓶頸在於硬碟的 I/O
 - ◆ 1 HDD = 75 MB/sec
- 解法: parallel reads
 - ◆ 1000 HDDs = 75 GB/sec

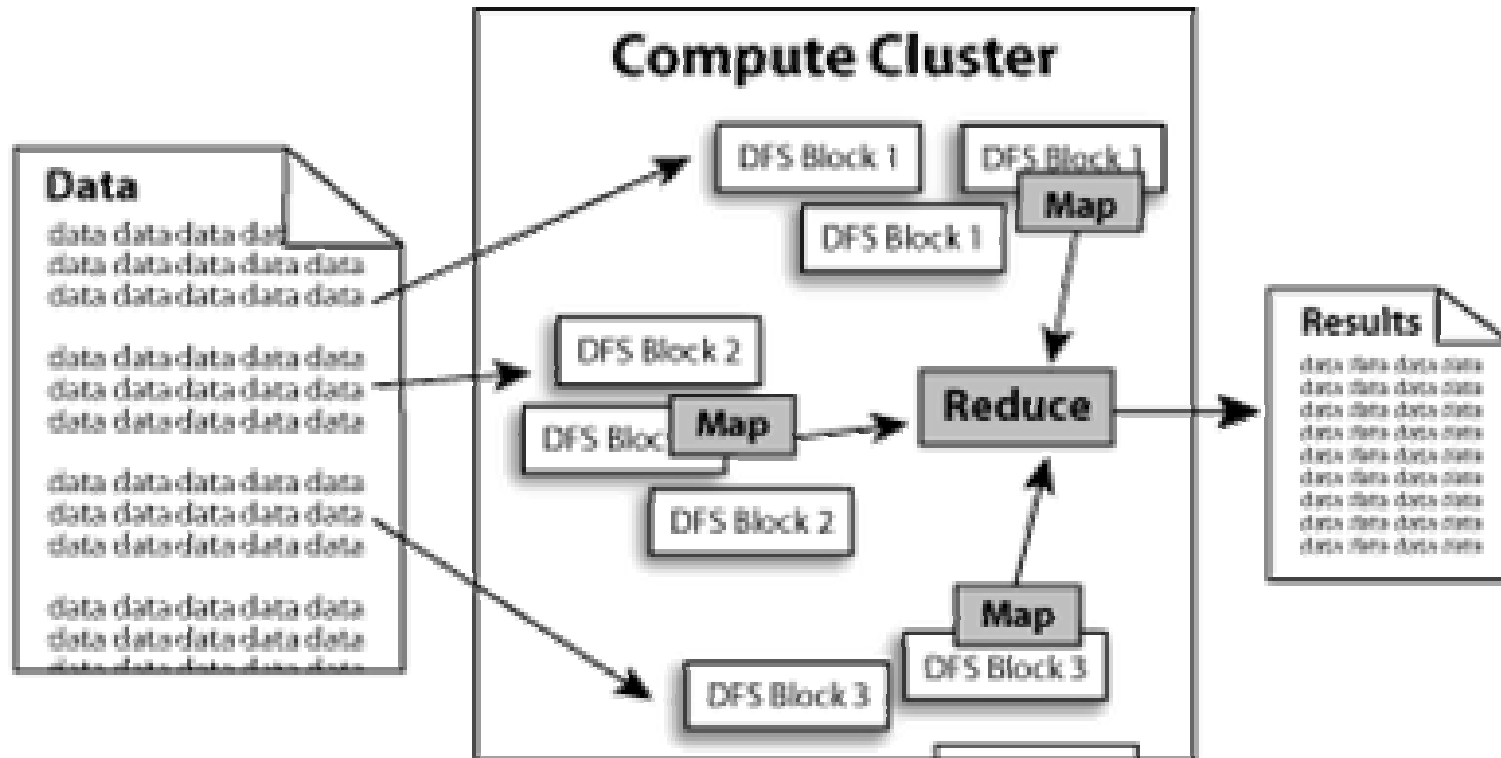
MapReduce 的動機

- Data > 1 TB
- 交互運算於大量的CPU
- 容易開發與使用
 - ◆ High-level applications written in MapReduce
 - ◆ Programmers don't worry about socket(), etc.

How MapReduce is Structured

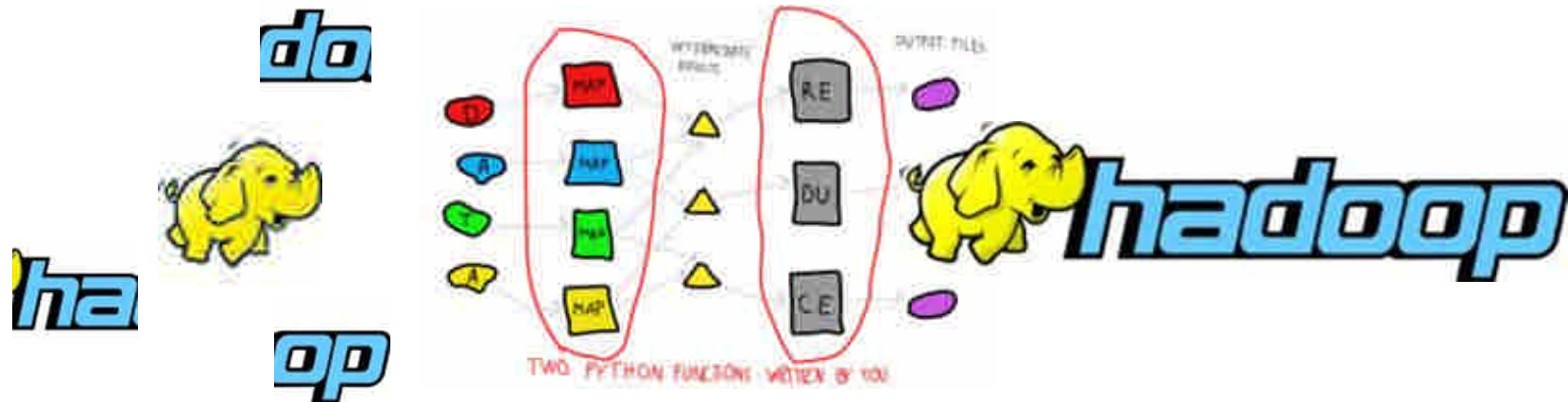
- 以Functional programming 架構分散式
運算
- 批次處理系統
- 可靠性

By Using Map / Reduce



MapReduce is a software framework to support distributed computing on large data sets on clusters of computers.

Hadoop MapReduce



Hadoop MapReduce 是一套儲存並處理 petabytes 等級資訊的雲端運算技術

Hadoop 提供

- Automatic parallelization & distribution
- Fault-tolerance
- Status and monitoring tools
- A clean abstraction and API for programmers

Hadoop Applications (1)

- Adknowledge - Ad network
 - ◆ behavioral targeting, clickstream analytics
- Alibaba
 - ◆ processing sorts of business data dumped out of database and joining them together. These data will then be fed into iSearch, our vertical search engine.
- Baidu - the leading Chinese language search engine
 - ◆ Hadoop used to analyze the log of search and do some mining work on web page database

Hadoop Applications (3)

- Facebook

- ◆ 處理 internal log and dimension data sources
- ◆ as a source for reporting/analytics and machine learning.

- Freestylers - Image retrieval engine

- ◆ use Hadoop 影像處理

- Hosting Habitat

- ◆ 取得所有clients的軟體資訊
- ◆ 分析並告知clients 未安裝或未更新的軟體

Hadoop Applications (4)

● IBM

- ◆ Blue Cloud Computing Clusters

● Journey Dynamics

- ◆ 用 Hadoop MapReduce 分析 billions of lines of GPS data 並產生交通路線資訊.

● Krugle

- ◆ 用 Hadoop and Nutch 建構 原始碼搜尋引擎

Hadoop Applications (5)

- SEDNS - Security Enhanced DNS Group

- ◆ 收集全世界的 DNS 以探索網路分散式內容。

- Technical analysis and Stock Research

- ◆ 分析股票資訊

- University of Maryland

- ◆ 用Hadoop執行 machine translation, language modeling, bioinformatics, email analysis, and image processing 相關研究

- University of Nebraska Lincoln, Research Computing Facility

- ◆ 用Hadoop跑約200TB的CMS經驗分析

- ◆ 緊湊渺子線圈（CMS，Compact Muon Solenoid）為瑞士歐洲核子研究組織CERN的大型強子對撞器計劃的兩大通用型粒子偵測器中的一個。

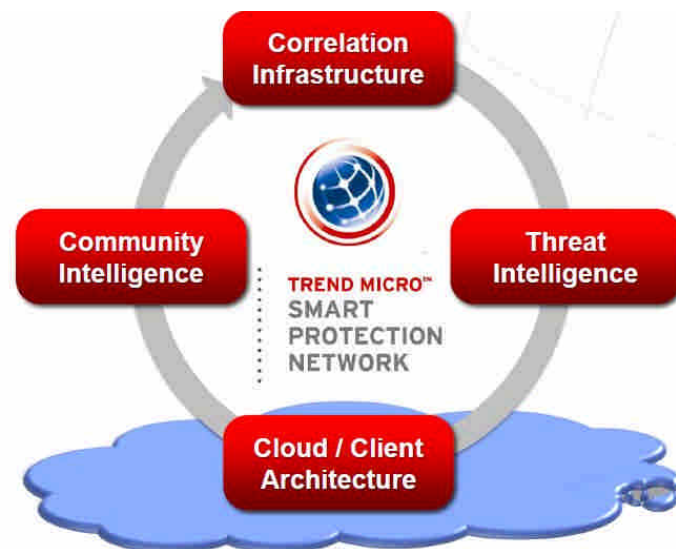
Hadoop Applications (6)

● Yahoo!

- ◆ Used to support research for Ad Systems and Web Search
- ◆ 使用Hadoop平台來發現發送垃圾郵件的殭屍網絡

● 趨勢科技

- ◆ 過濾像是釣魚網站或惡意連結的網頁內容



MapReduce Conclusions

- 適用於 large-scale applications 以及 large-scale computations
- 程式設計者只需要解決”真實的”問題，架構面留給MapReduce
- MapReduce可應用於多種領域：
 - ◆ Text tokenization, Indexing and Search, Data mining, machine learning...

Map Reduce

原理說明

© TemplatesW.com

王耀聰 陳威宇

Jazz@nchc.org.tw

waue@nchc.org.tw



財團法人國家實驗研究院

國家高速網路與計算中心

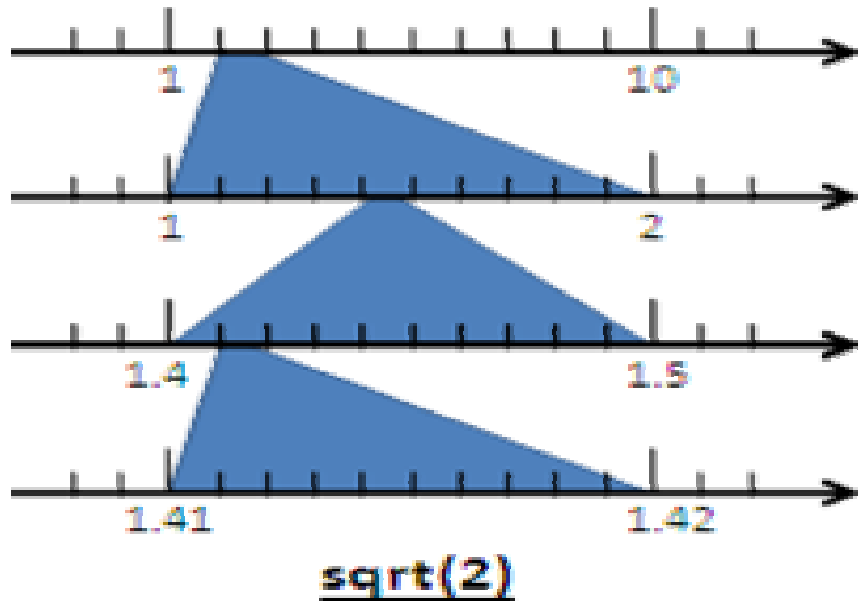
NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING

Algorithms

- Functional Programming : Map Reduce
 - ◆ `map(...)` :
 - `[1,2,3,4] - (*2) -> [2,4,6,8]`
 - ◆ `reduce(...)`:
 - `[1,2,3,4] - (sum) -> 10`
 - ◆ 對應演算法中的Divide and conquer
 - ◆ 將問題分解成很多個小問題之後，再做總和

Divide and Conquer

範例一：十分逼近法



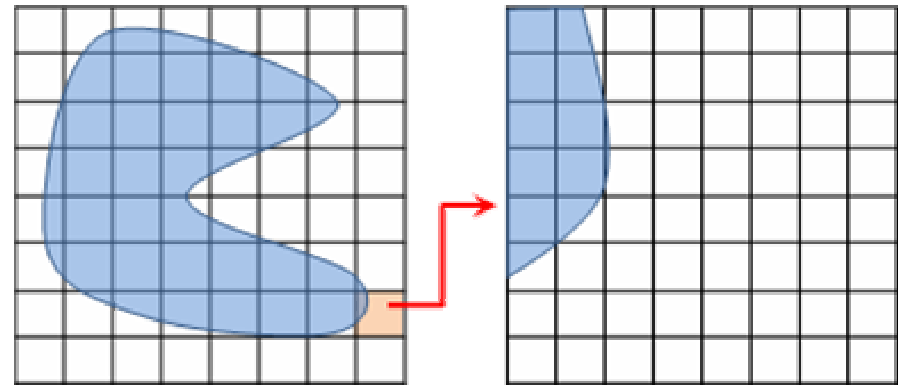
範例四：

眼前有五階樓梯，每次可踏上一階或踏上兩階，那麼爬完五階共有幾種踏法？

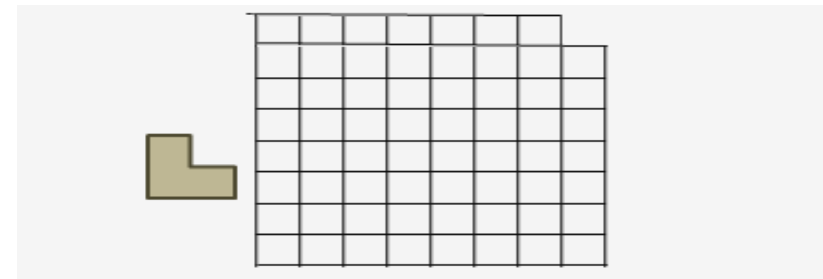
Ex : (1, 1, 1, 1, 1) or

(1, 2, 1, 1)

範例二：方格法求面積



範例三：鋪滿 L 形磁磚



Programming Model

- Users implement interface of two functions:
 - ◆ $\text{map}(\text{in_key}, \text{in_value}) \rightarrow (\text{out_key}, \text{intermediate_value}) \text{ list}$
 - ◆ $\text{reduce}(\text{out_key}, \text{intermediate_value list}) \rightarrow \text{out_value list}$

Map

● One-to-one Mapper

```
let map(k, v) =  
Emit(k.toUpper(),  
v.toUpper())
```

(“Foo”, “other”) → (“FOO”, “OTHER”)
 (“key2”, “data”) → (“KEY2”, “DATA”)

● Explode Mapper

```
let map(k, v) =  
foreach char c in v:  
emit(k, c)
```

(“A”, “cats”) → (“A”, “c”), (“A”, “a”),
 (“A”, “t”), (“A”, “s”)

● Filter Mapper

```
let map(k, v) =  
if (isPrime(v))  
then emit(k, v)
```

(“foo”, 7) → (“foo”, 7)
 (“test”, 10) → (nothing)

Reduce

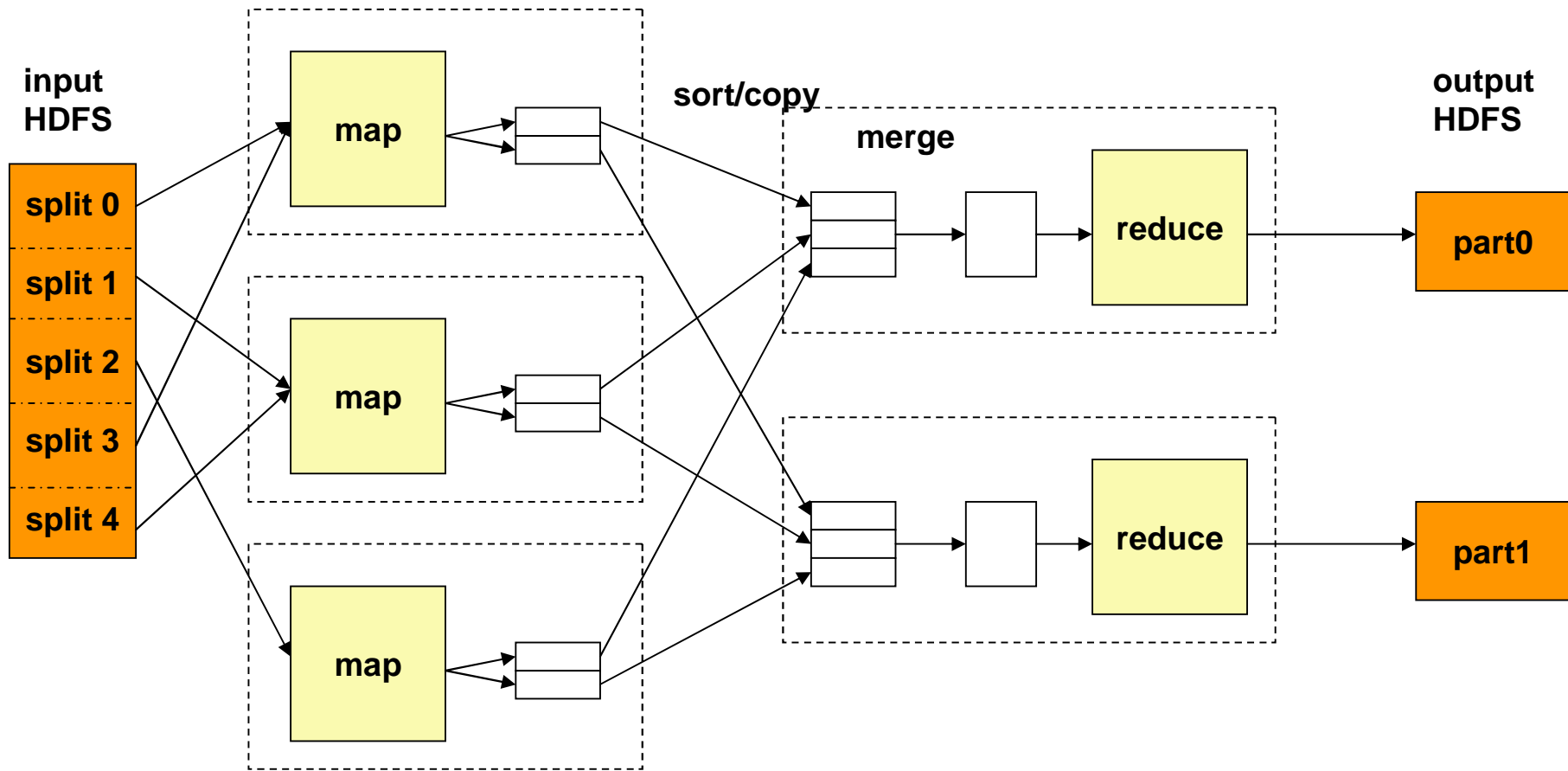
Example: Sum Reducer

```
let reduce(k, vals) =  
  sum = 0  
  foreach int v in vals:  
    sum += v  
  emit(k, sum)
```

(“A”, [42, 100, 312]) → (“A”, 454)

(“B”, [12, 6, -2]) → (“B”, 16)

MapReduce 運作流程



JobTracker跟NameNode取得需要運算的blocks

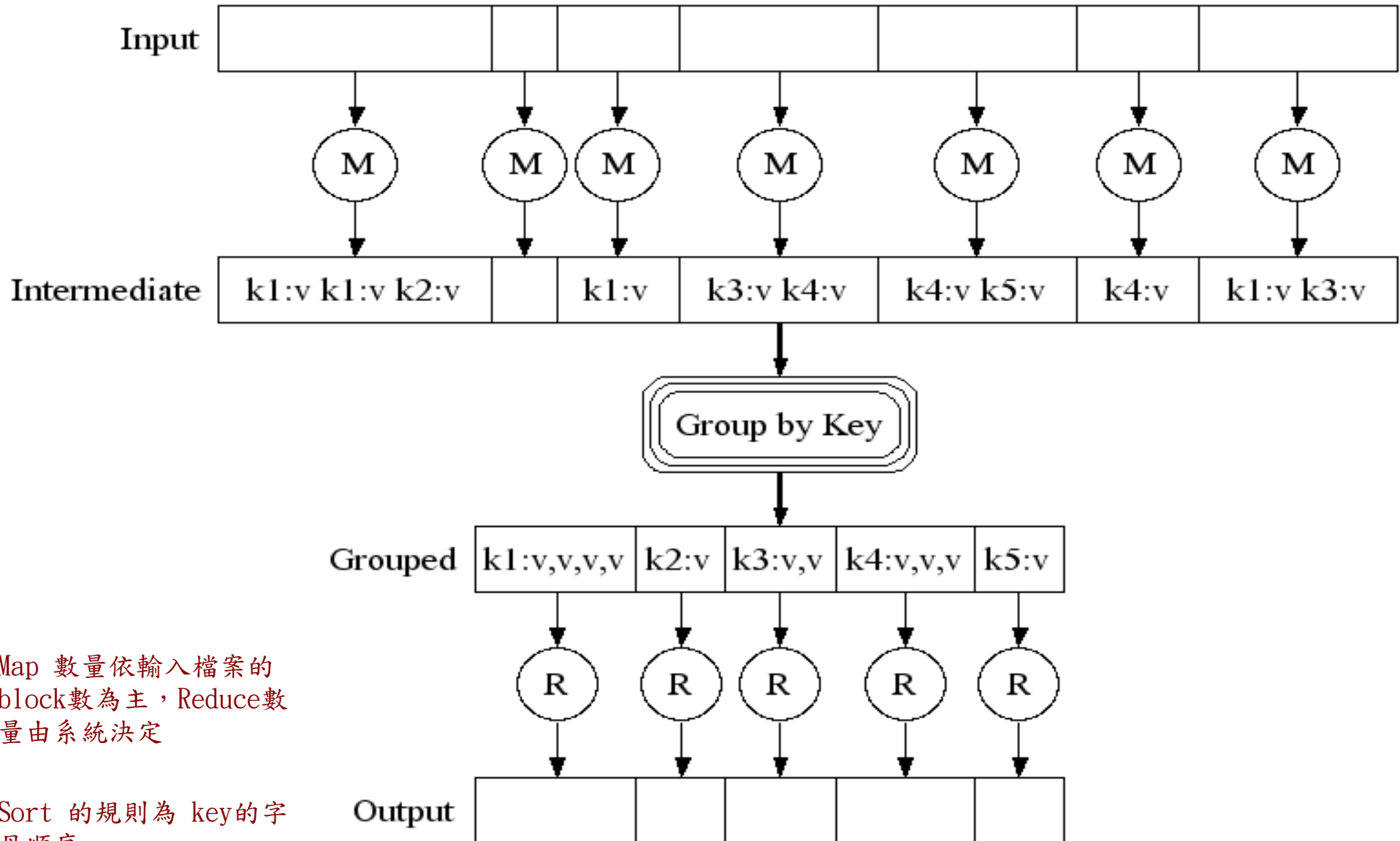
JobTracker選數個TaskTracker來作Map運算，產生些中間檔案

JobTracker將中間檔案整合排序後，複製到需要的TaskTracker去

JobTracker派遣TaskTracker作reduce

reduce完後通知JobTracker與NameNode以產生output

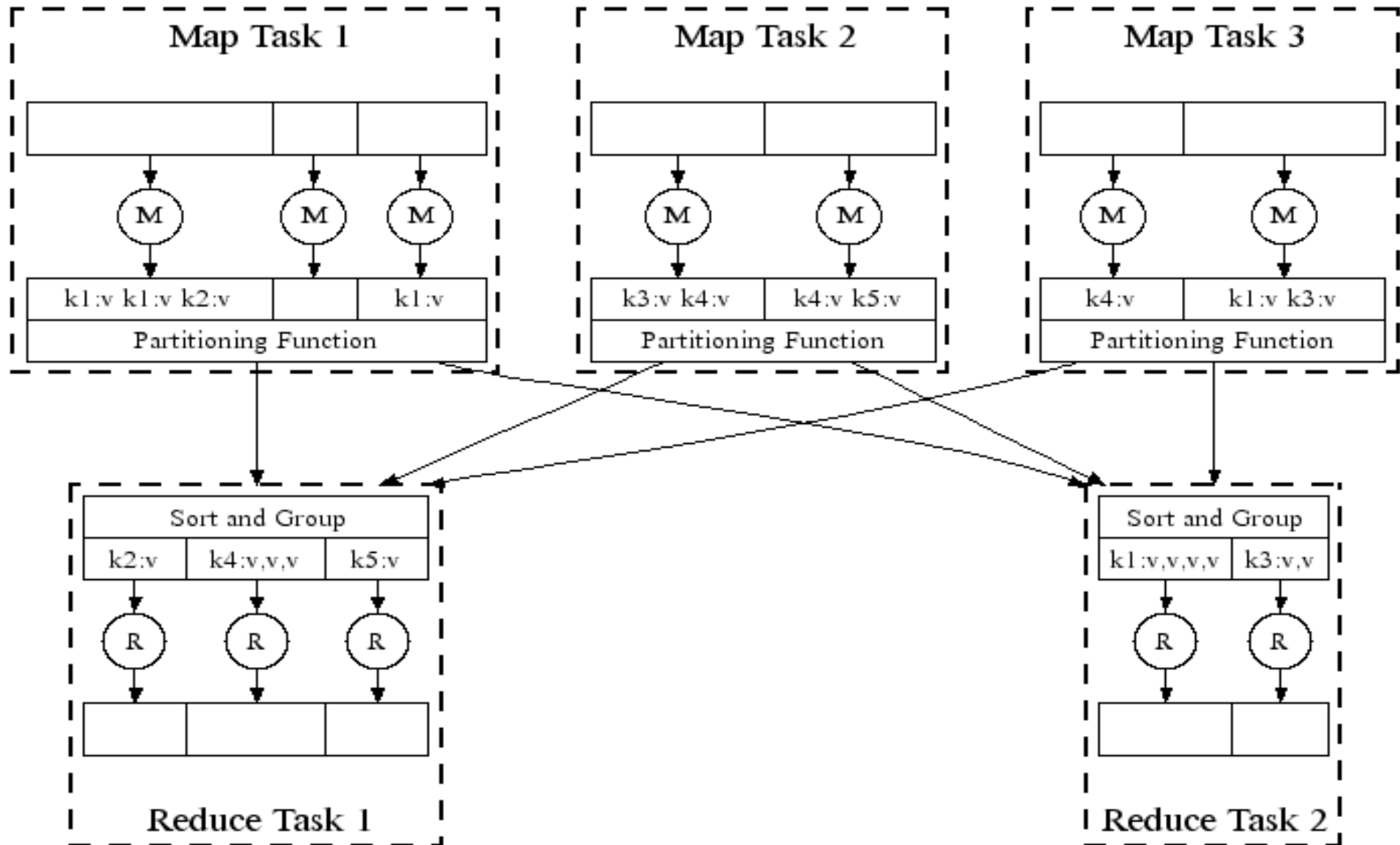
MapReduce 圖解



Map 數量依輸入檔案的 block 數為主，Reduce 數量由系統決定

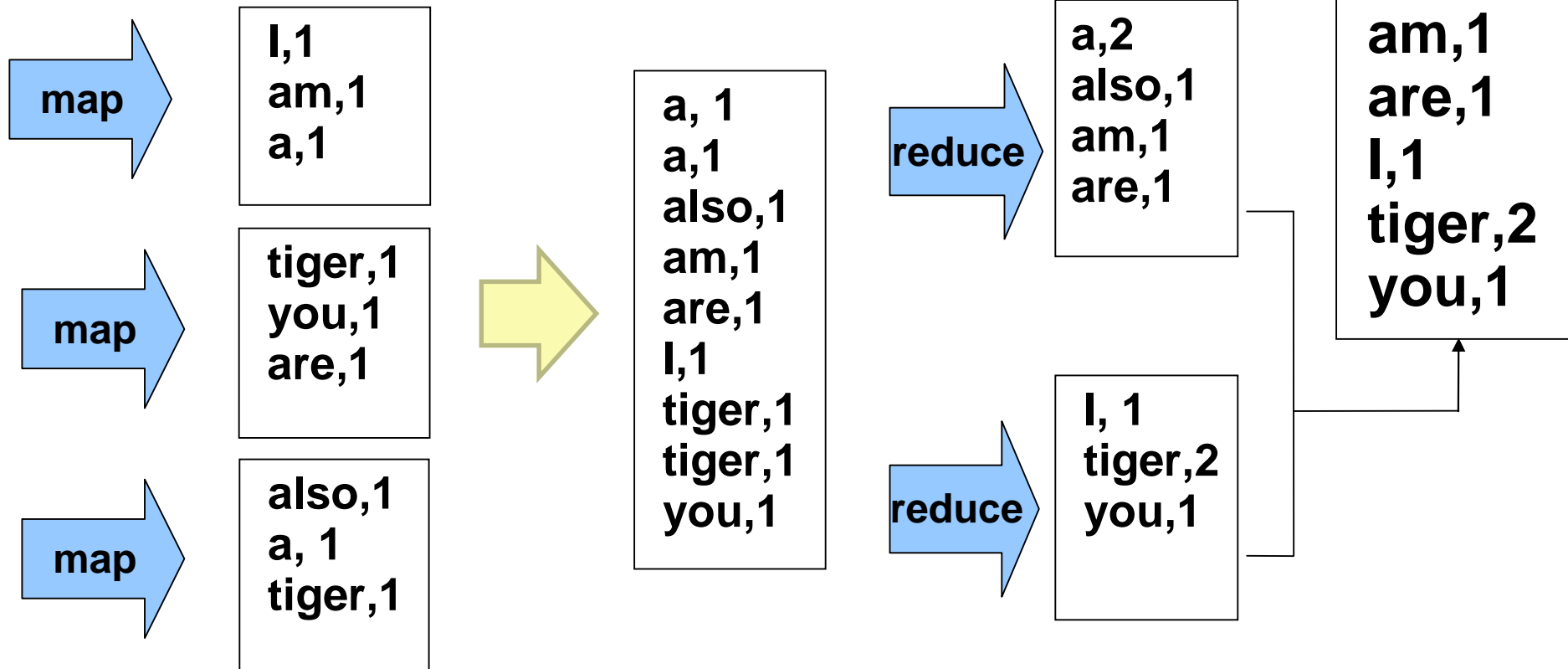
Sort 的規則為 key 的字母順序

MapReduce in Parallel



範例

I am a tiger, you are also a tiger



JobTracker先選了三個 Tracker做map

Map結束後，hadoop進行中間資料的整理與排序

JobTracker再選兩個 TaskTracker作reduce

Console 端 編譯與執行

王耀聰 陳威宇

Jazz@nchc.org.tw

waue@nchc.org.tw



財團法人國家實驗研究院

國家高速網路與計算中心

NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING

Java 之編譯與執行

1. 編譯

◆ `javac` Δ `-classpath` Δ `hadoop-*-core.jar` Δ `-d` Δ `MyJava`
 Δ `MyCode.java`

2. 封裝

◆ `jar` Δ `-cvf` Δ `MyJar.jar` Δ `-C` Δ `MyJava` Δ `.`

3. 執行

◆ `bin/hadoop` Δ `jar` Δ `MyJar.jar` Δ `MyCode` Δ
`HDFS_Input/` Δ `HDFS_Output/`

-
- 所在的執行目錄為Hadoop_Home
 - `./MyJava` = 編譯後程式碼目錄
 - `My jar. jar` = 封裝後的編譯檔

- 先放些文件檔到HDFS上的input目錄
- `./input`; `./ouput` = hdfs的輸入、輸出目錄

WordCount1 練習 (I)

1. `cd $HADOOP_HOME`
2. `bin/hadoop dfs -mkdir input`
3. `echo "I like NCHC Cloud Course." > inputwc/input1`
4. `echo "I like nchc Cloud Course, and we enjoy this crouse." > inputwc/input2`
5. `bin/hadoop dfs -put inputwc inputwc`
6. `bin/hadoop dfs -ls input`

```
waue@vPro:/opt/hadoop$ bin/hadoop dfs -ls input
Found 2 items
-rw-r--r--  1 waue supergroup    26 2009-03-22 12:15 /user/waue/input/input1
-rw-r--r--  1 waue supergroup    52 2009-03-22 12:15 /user/waue/input/input2
waue@vPro:/opt/hadoop$
```

WordCount1 練習 (II)

1. 編輯 WordCount.java

http://trac.nchc.org.tw/cloud/attachment/wiki/jazz/Hadoop_Lab6/WordCount.java?format=raw

2. mkdir MyJava

3. javac -classpath hadoop-*-core.jar -d MyJava
WordCount.java

4. jar -cvf wordcount.jar -C MyJava .

5. bin/hadoop jar wordcount.jar WordCount input/ output/

-
- 所在的執行目錄為Hadoop_Home (因為hadoop-*-core.jar)
 - javac編譯時需要classpath, 但hadoop jar時不用
 - wordcount.jar = 封裝後的編譯檔, 但執行時需告知class name
 - Hadoop進行運算時, 只有 input 檔要放到hdfs上, 以便hadoop分析運算; 執行檔 (wordcount.jar) 不需上傳, 也不需每個node都放, 程式的載入交由java處理

WordCount1 練習 (III)

```
waue@vPro:/opt/hadoop$ mkdir MyJava
waue@vPro:/opt/hadoop$ javac -classpath hadoop-*-core.jar -d MyJava WordCount.java
waue@vPro:/opt/hadoop$ jar -cvf wordcount.jar -C MyJava .
新增 manifest
新增 : WordCount.class (讀=1516)(寫=740)(壓縮 51%)
新增 : WordCount$Reduce.class (讀=1591)(寫=642)(壓縮 59%)
新增 : WordCount$Map.class (讀=1918)(寫=795)(壓縮 58%)
waue@vPro:/opt/hadoop$ bin/hadoop jar wordcount.jar WordCount input/ output/
09/03/22 11:39:01 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
09/03/22 11:39:01 INFO mapred.FileInputFormat: Total input paths to process : 1
09/03/22 11:39:01 INFO mapred.FileInputFormat: Total input paths to process : 1
09/03/22 11:39:02 INFO mapred.JobClient: Running job: job_200903201526_0007
09/03/22 11:39:03 INFO mapred.JobClient: map 0% reduce 0%
09/03/22 11:39:08 INFO mapred.JobClient: map 100% reduce 0%
09/03/22 11:39:15 INFO mapred.JobClient: Job complete: job_200903201526_0007
09/03/22 11:39:15 INFO mapred.JobClient: Counters: 16
09/03/22 11:39:15 INFO mapred.JobClient:   File Systems
09/03/22 11:39:15 INFO mapred.JobClient:     HDFS bytes read=320950
09/03/22 11:39:15 INFO mapred.JobClient:     HDFS bytes written=130568
09/03/22 11:39:15 INFO mapred.JobClient:     Local bytes read=168448
09/03/22 11:39:15 INFO mapred.JobClient:     Local bytes written=336932
09/03/22 11:39:15 INFO mapred.JobClient:   Job Counters
09/03/22 11:39:15 INFO mapred.JobClient:     Launched reduce tasks=1
```


WordCount1 練習(IV)

```
waue@vPro:/opt/hadoop$ bin/hadoop dfs -cat output/part-00000
Cloud      2
Course,    1
Course.    1
I          2
NCHC      1
and        1
course.    1
enjoy      1
like       2
nchc       1
this       1
we         1
```

BTW ...

- 雖然Hadoop框架是用Java實作，但Map/Reduce應用程序則不一定要用Java來寫
- Hadoop Streaming：
 - ◆ 執行作業的工具，使用者可以用其他語言（如：PHP）套用到Hadoop的mapper和reducer
- Hadoop Pipes：C++ API

透過Eclipse開發

© TemplatesW.com

王耀聰 陳威宇

Jazz@nchc.org.tw

waue@nchc.org.tw



財團法人國家實驗研究院

國家高速網路與計算中心

NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING

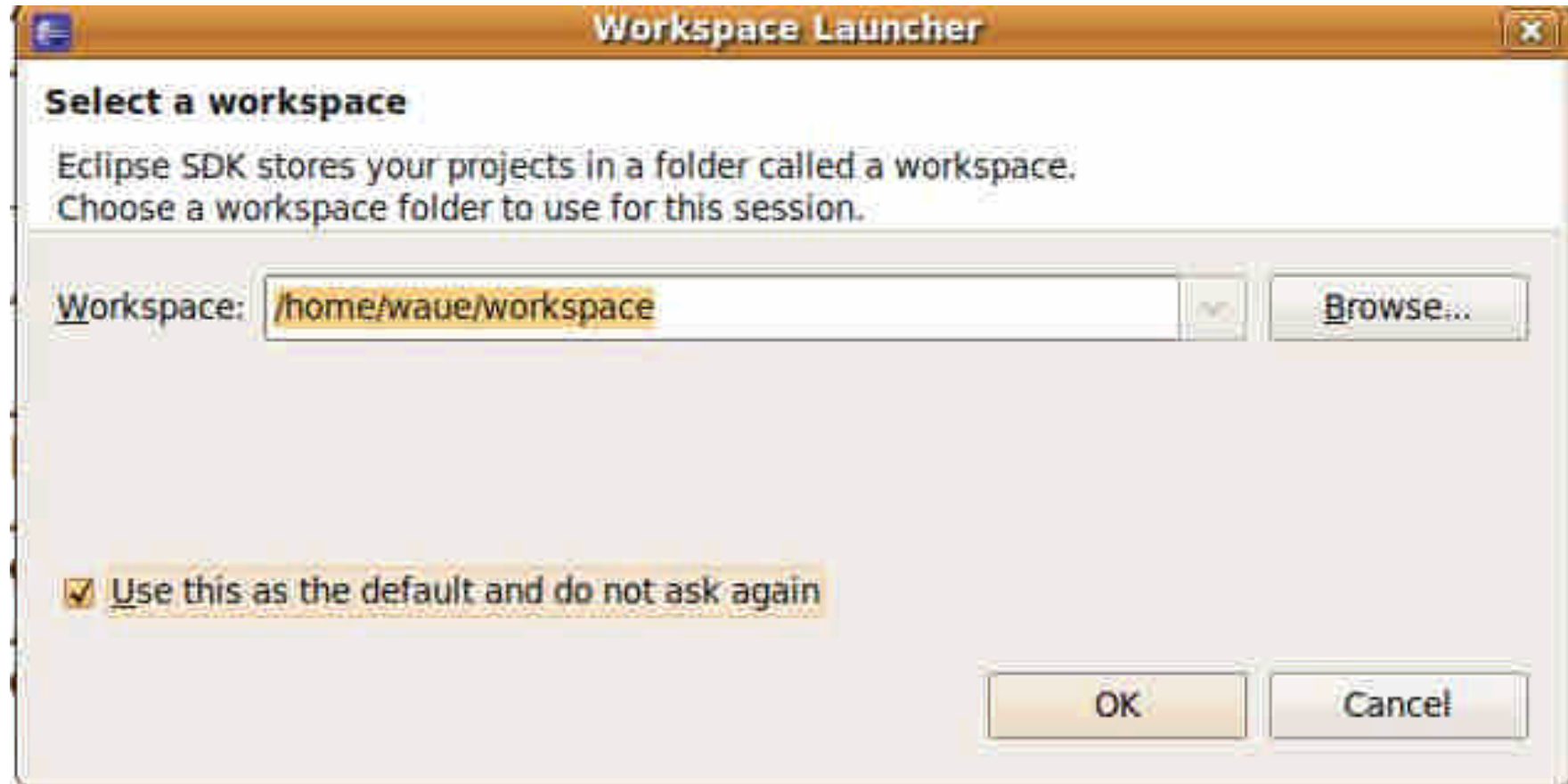
Requirements

- Hadoop 0.20.0 up
- Java 1.6
- Eclipse 3.3 up
- Hadoop Eclipse Plugin 0.20.0 up

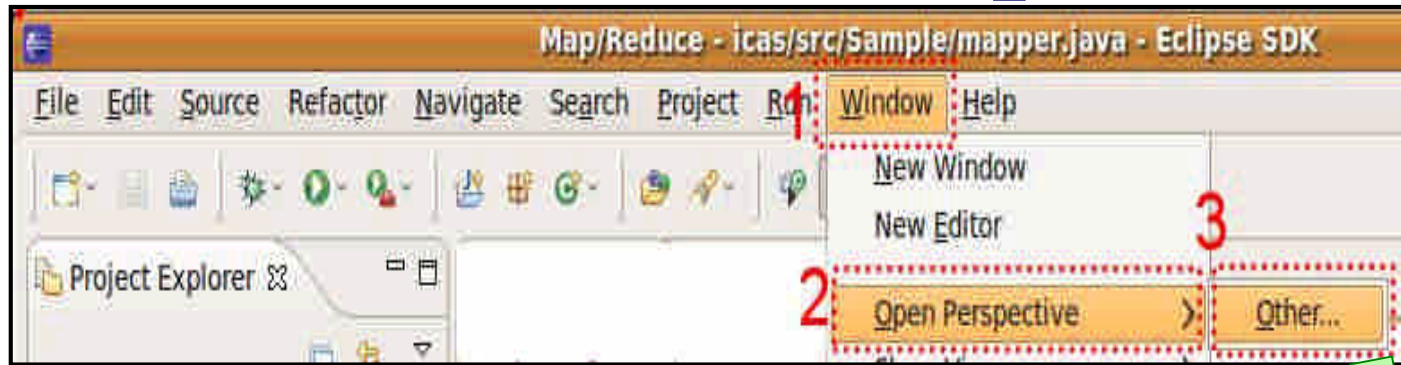
安裝 Hadoop Eclipse Plugin

- Hadoop Eclipse Plugin 0.20.0
 - ◆ From
\$Hadoop_0.20.0_home/contrib/eclipse-plugin/hadoop-0.20.0-eclipse-plugin.jar
- Hadoop Eclipse Plugin 0.20.1
 - ◆ Compiler needed
 - ◆ Or download from
<http://hadoop-eclipse-plugin.googlecode.com/files/hadoop-0.20.1-eclipse-plugin.jar>
- copy to **\$Eclipse_home/plugins/**

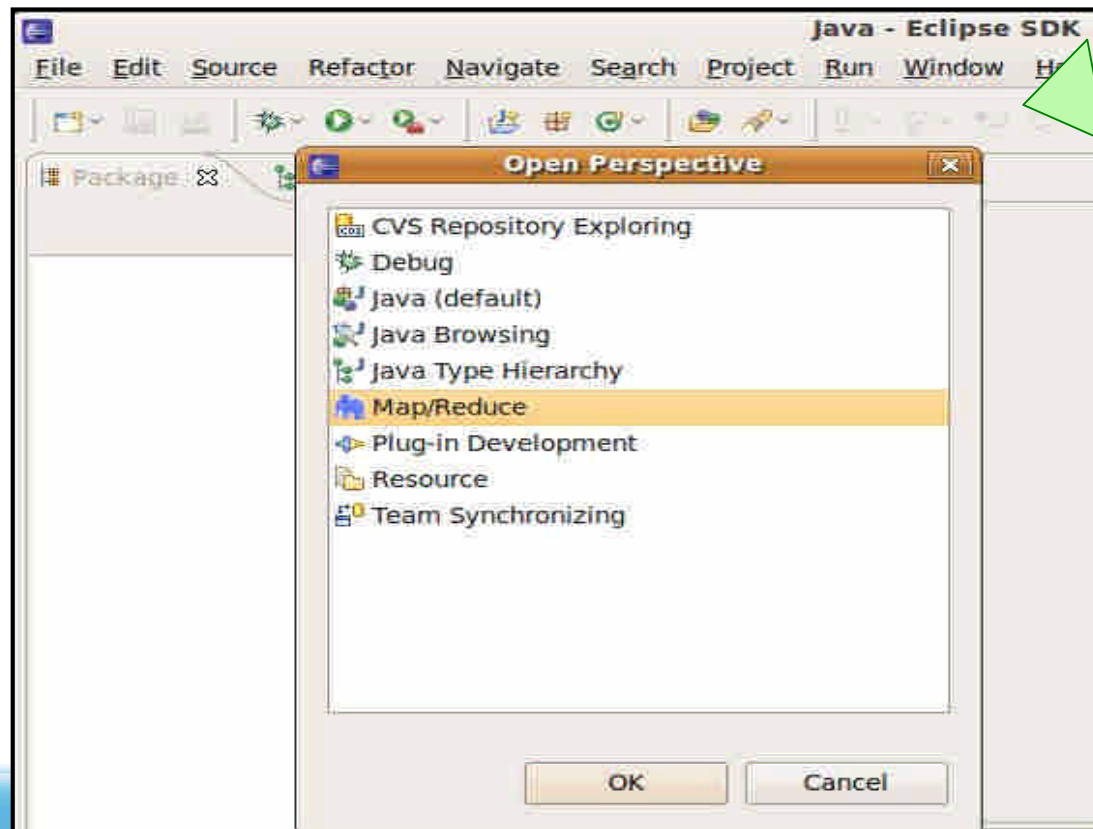
1 打開Eclipse, 設定專案目錄



2. 使用 Hadoop mode 視野

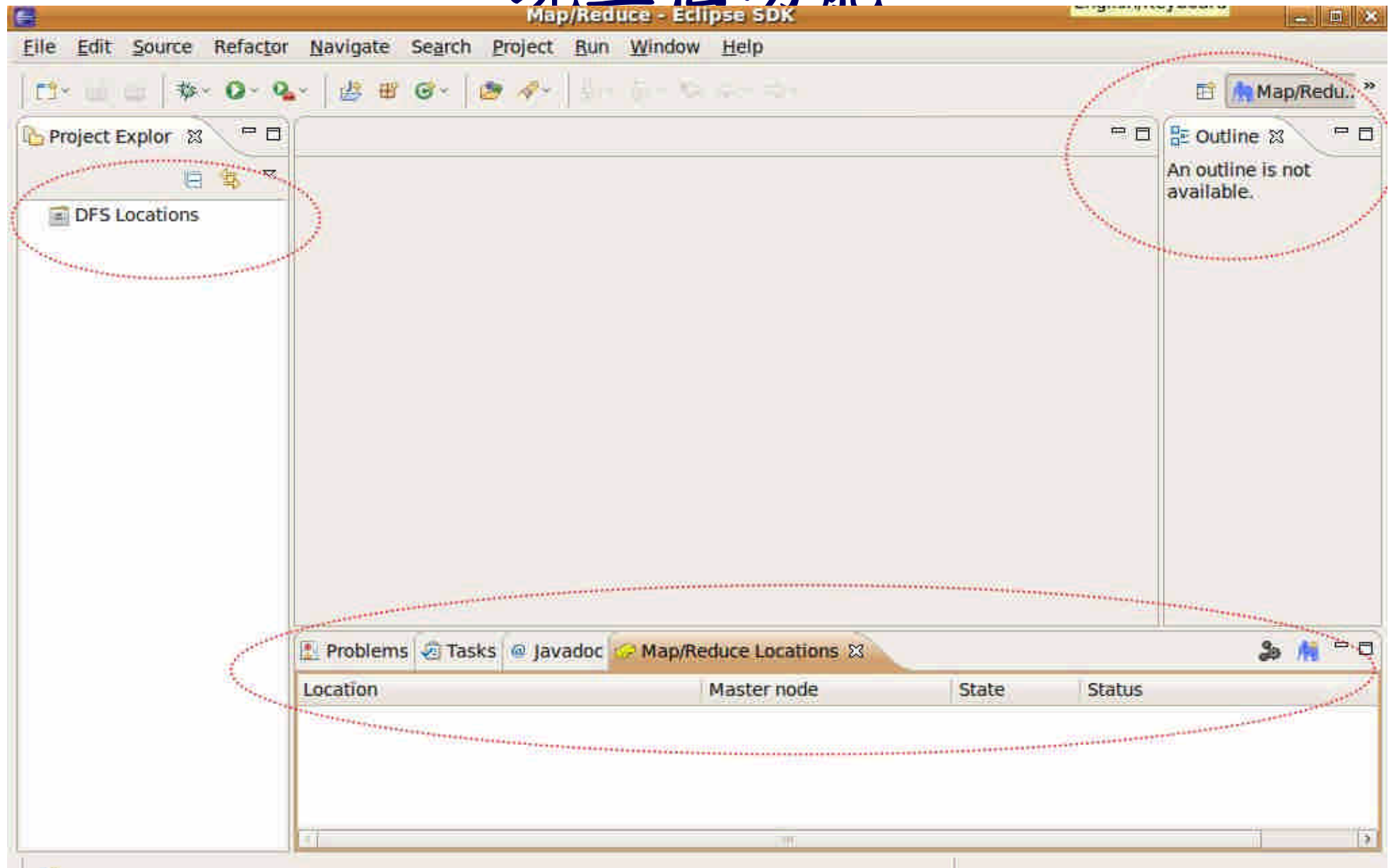


Window →
Open Perspective
→ Other

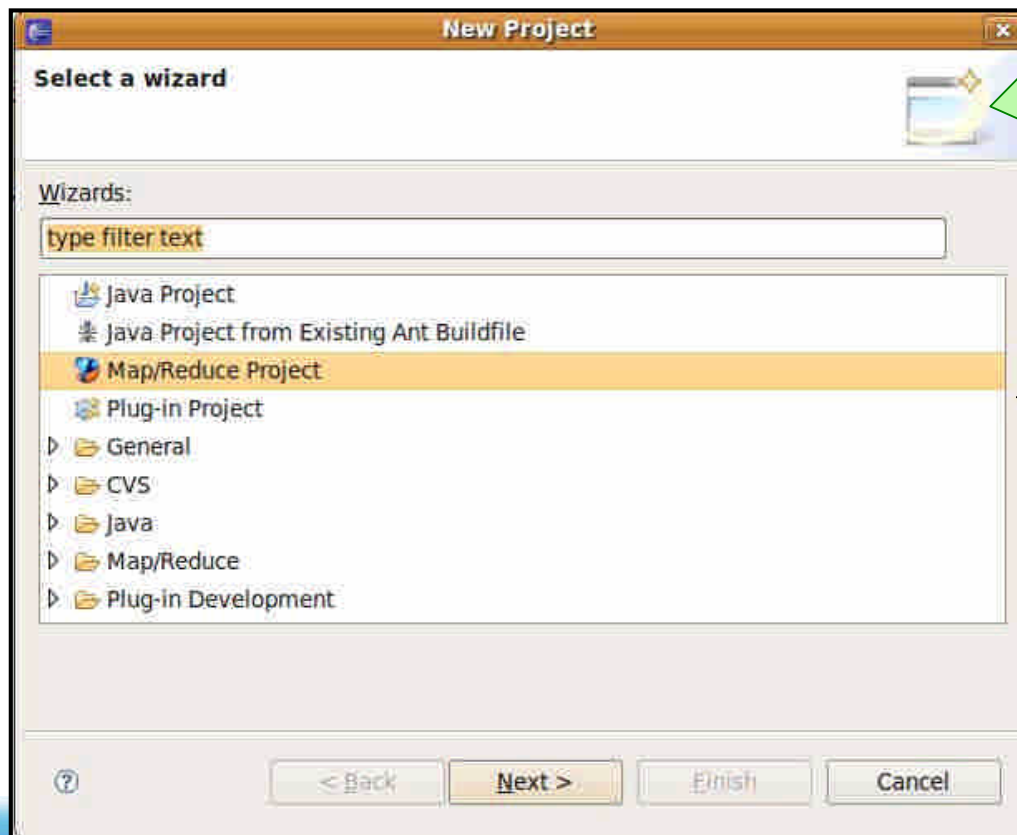
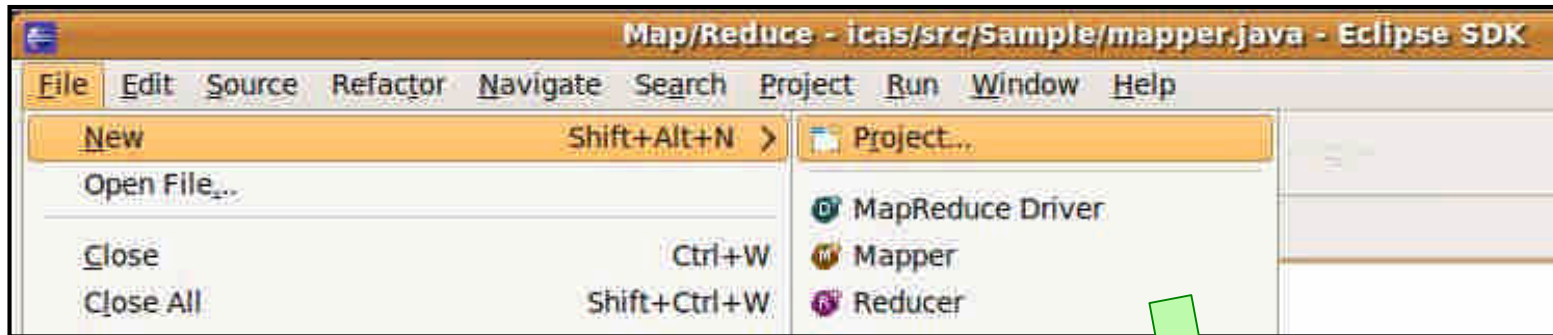


若有看到
MapReduce的大象
圖示代表Hadoop
Eclipse plugin
有安裝成功，若
沒有請檢查是否有
安之裝正確

3. 使用Hadoop視野，主畫面將出現三個功能



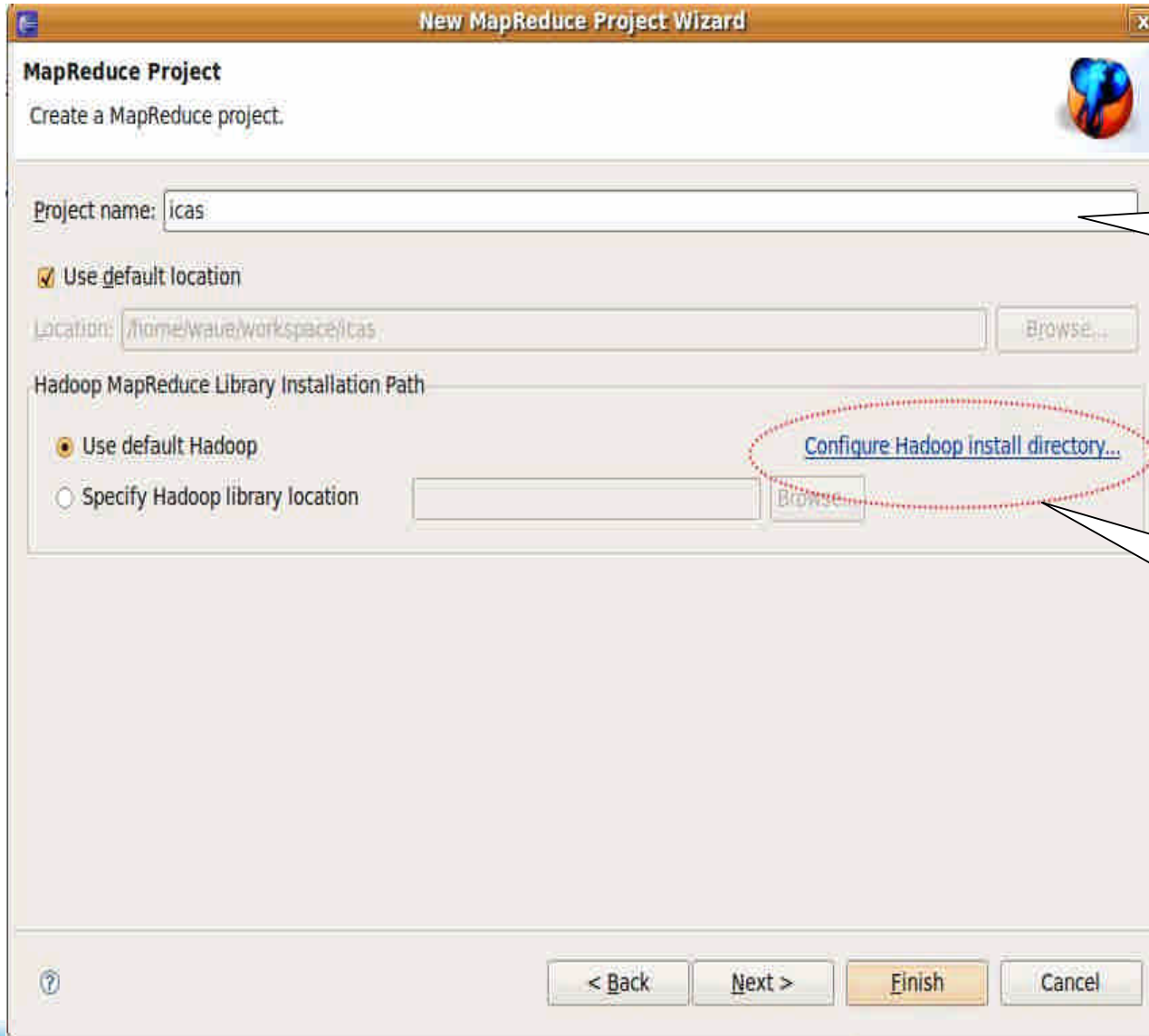
4. 建立一個Hadoop專案



開出新專案

選擇Map/Reduce
專案

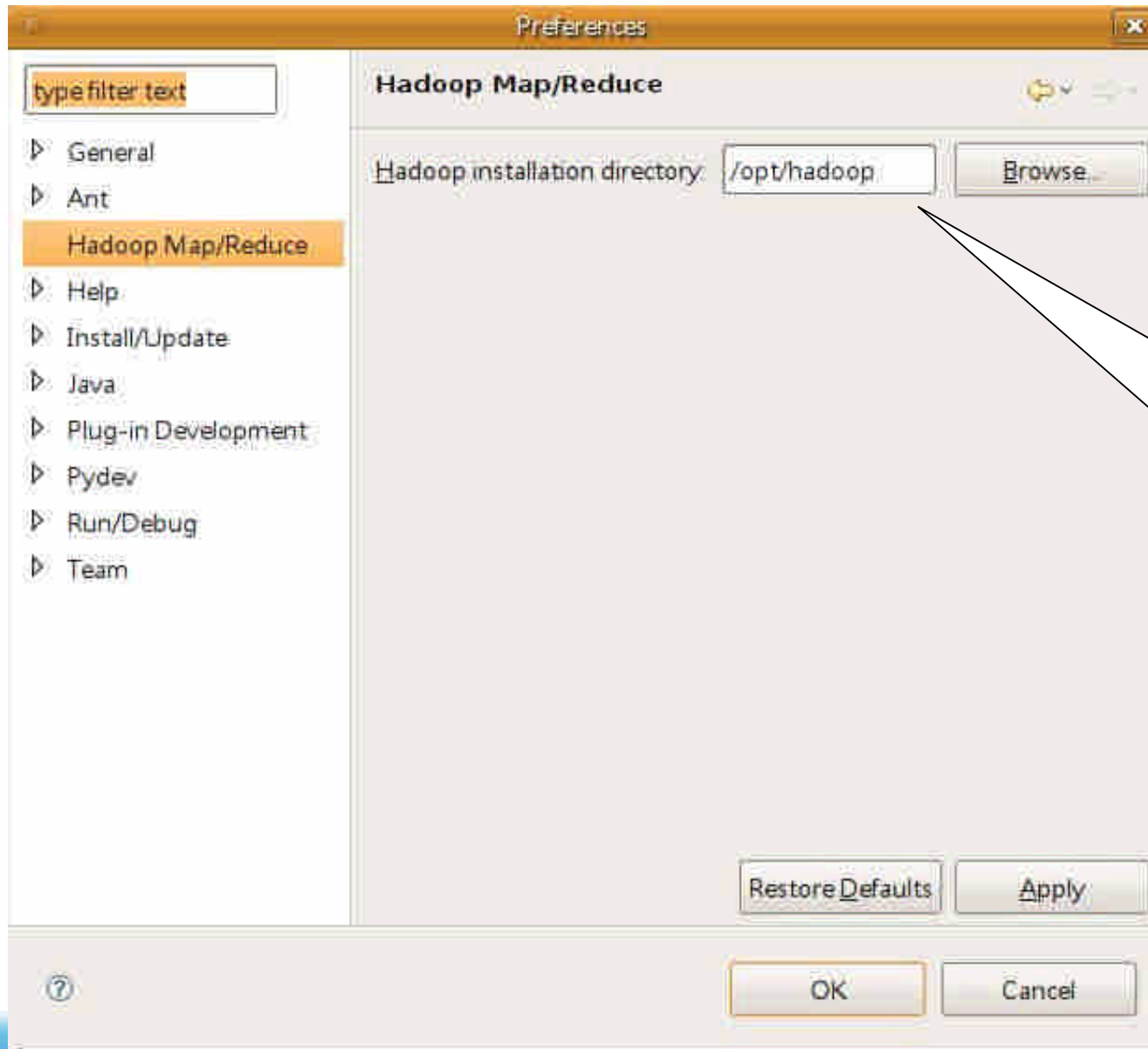
4-1. 輸入專案名稱並點選設定 Hadoop安裝路徑



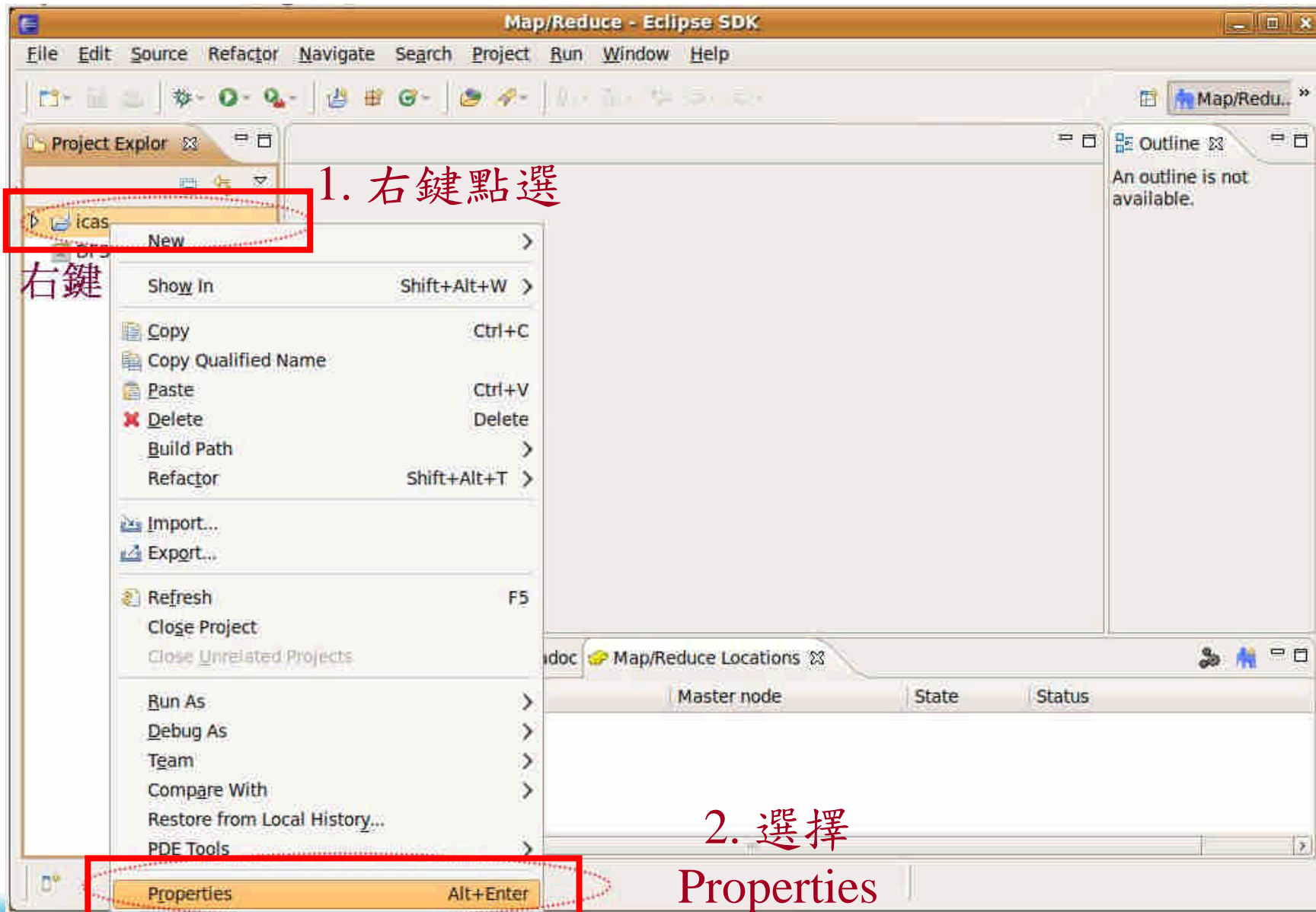
由此設定
專案名稱

由此設定
Hadoop的
安裝路徑

4-1-1. 填入Hadoop安裝路徑



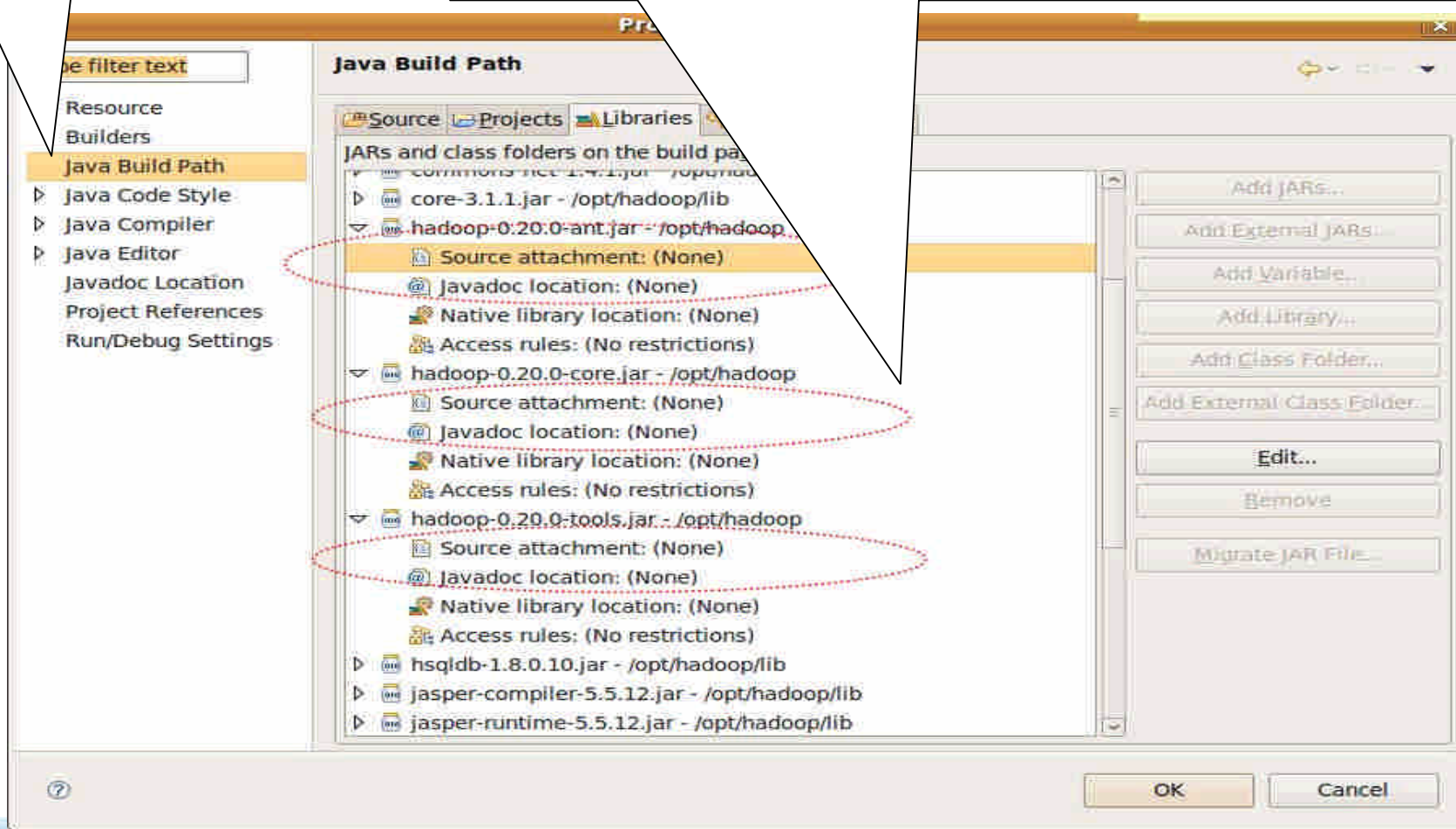
5. 設定Hadoop專案細節



5-1. 設定原始碼與文件路徑

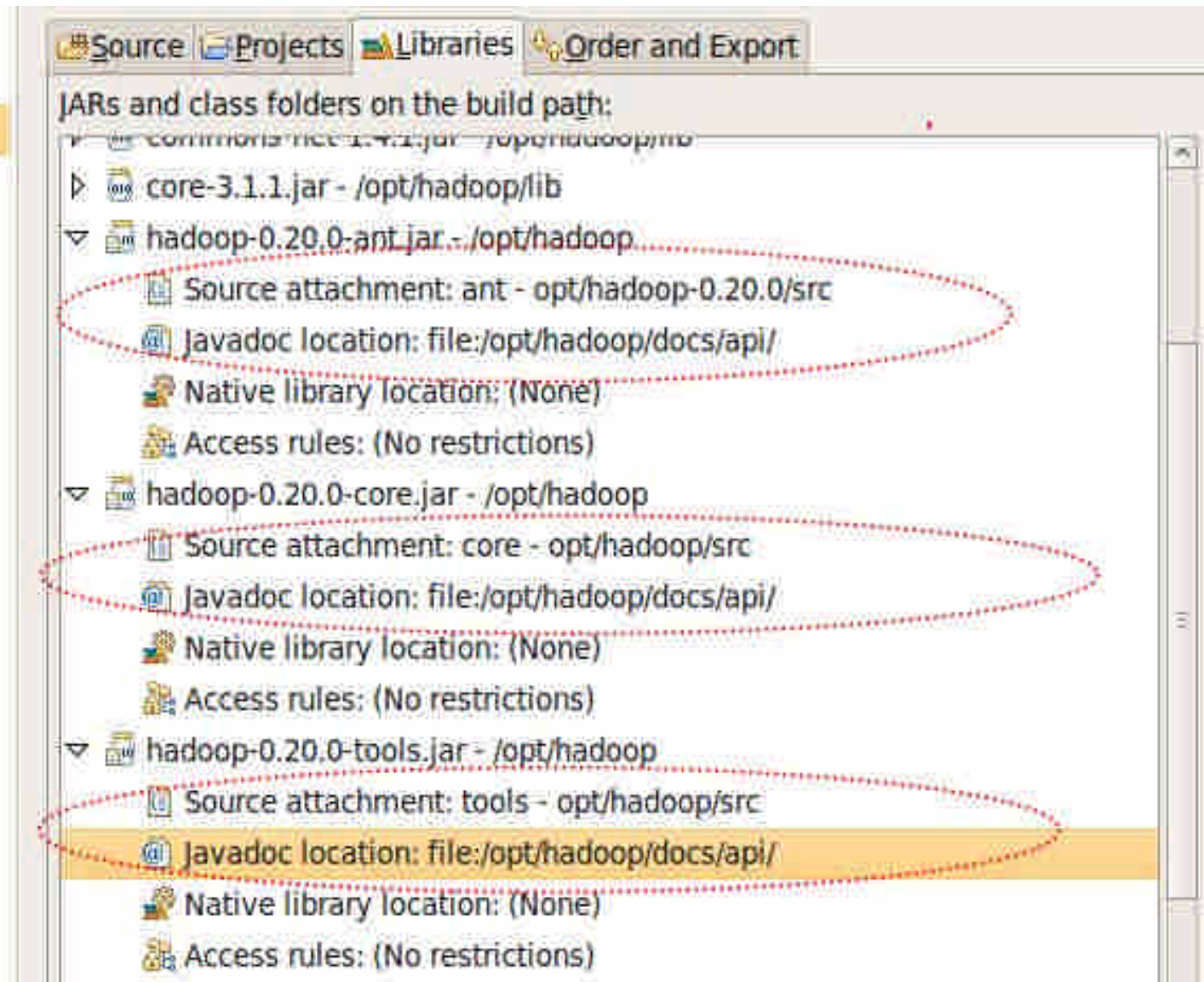
選擇 Java Build Path

以下請輸入正確的Hadoop原始碼與API文件檔路徑，如
source : /opt/hadoop/src/core/
javadoc : file:/opt/hadoop/docs/api/



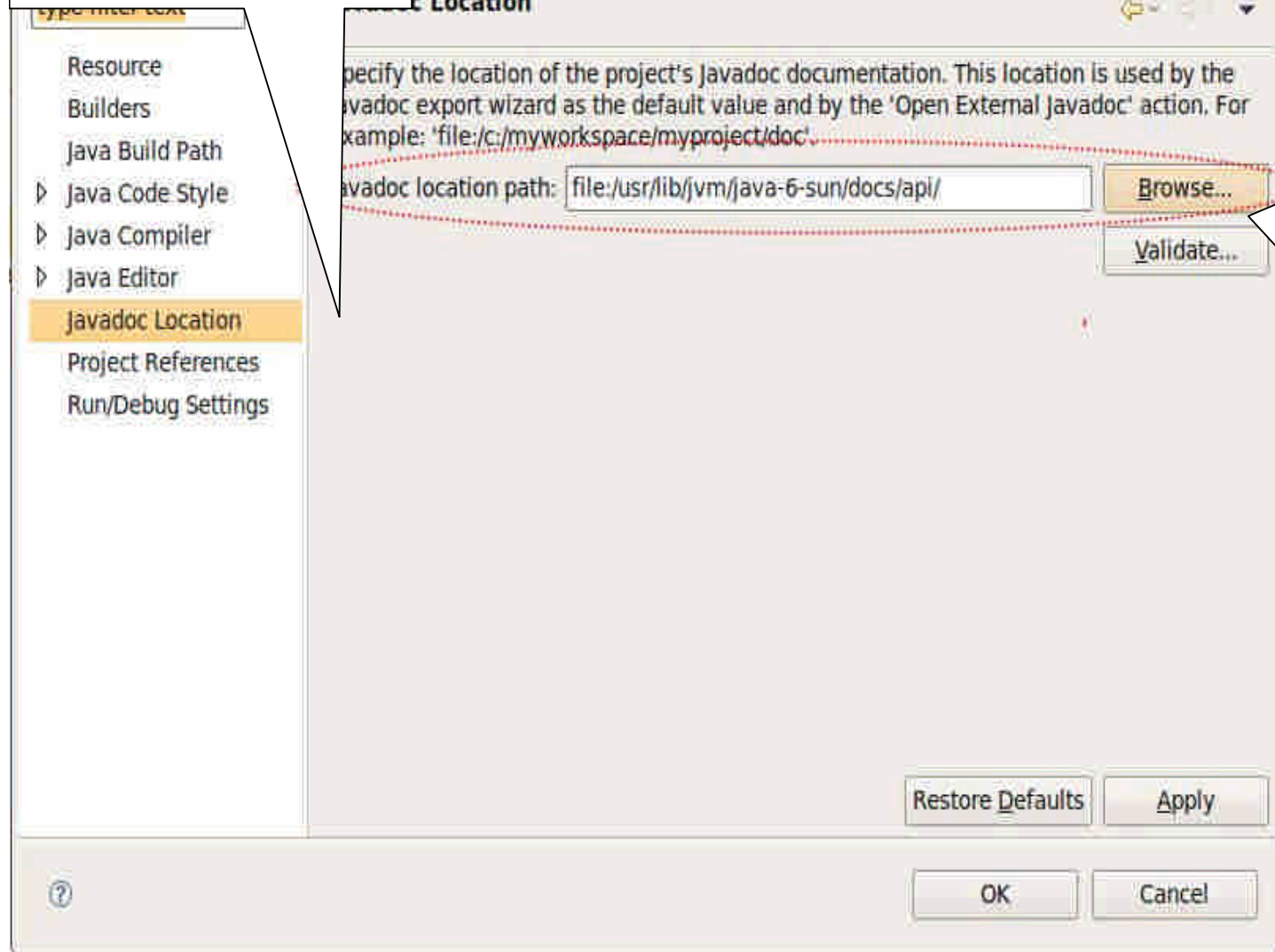
5-1-1. 完成圖

- Resource
- Builders
- Java Build Path**
- ▶ Java Code Style
- ▶ Java Compiler
- ▶ Java Editor
- Javadoc Location
- Project References
- Run/Debug Settings



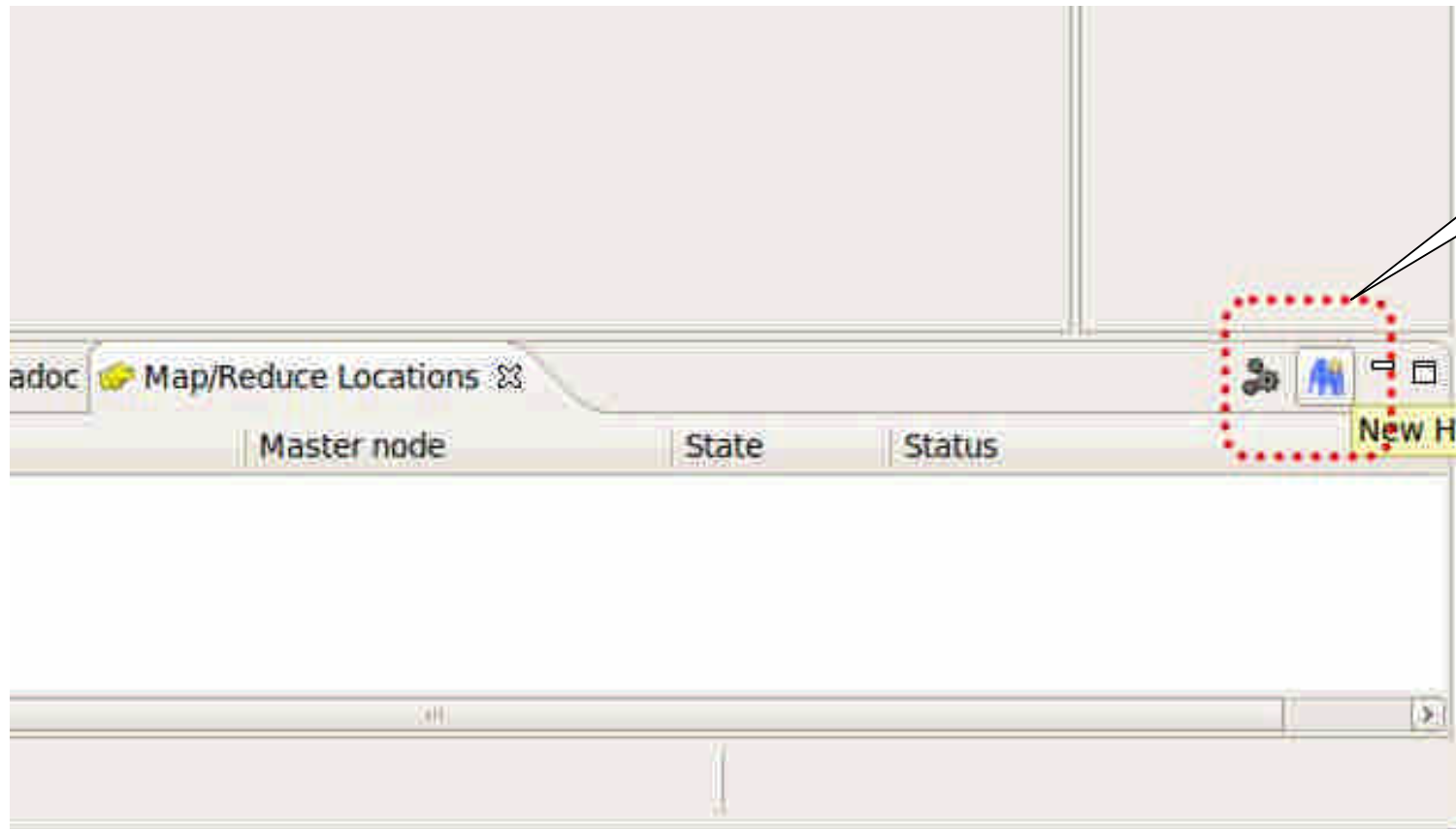
5-2. 設定java doc的完整路徑

選擇 Javadoc Location



輸入java 6 的 API正確路徑，輸入完後可選擇validate以驗證是否正確

6. 連結Hadoop Server與Eclipse



6-1. 設定你要連接的Hadoop主機

任意填一個名稱

輸入主機位址或 domain name

MapReduce 監聽的 Port (設定於mapred-site.xml)

HDFS 監聽的 Port (設定於core-site.xml)

你在此 Hadoop Server 上的 Username

Define Hadoop location

Define the location of a Hadoop infrastructure for running MapReduce applications.

General | Advanced parameters

Location name:

Map/Reduce Master

Host:

Port:

DFS Master

Use M/R Master host

Host:

Port:

User name:

SOCKS-proxy

Enable SOCKS proxy

Host:

Port:

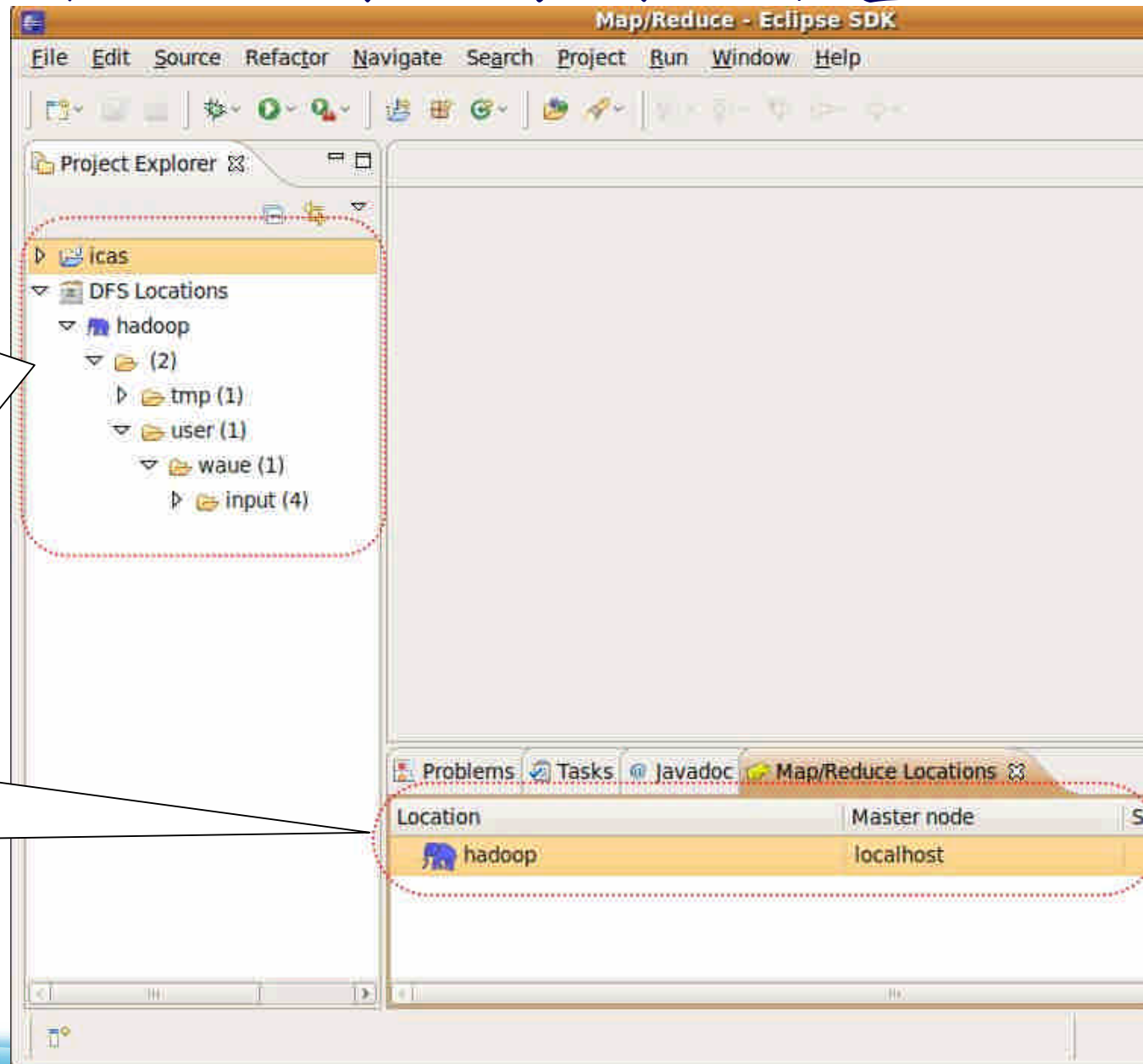
Load from file | Validate location

Finish | Cancel

6-2 若正確設定則可得到以下畫面

HDFS的資訊，可直接於此操作檢視、新增、上傳、刪除等命令

若有Job運作，可於此視窗檢視



7. 新增一個Hadoop程式

首先先建立一個 WordCount 程式，其他欄位任意

7.1 於程式窗格內輸入程式碼

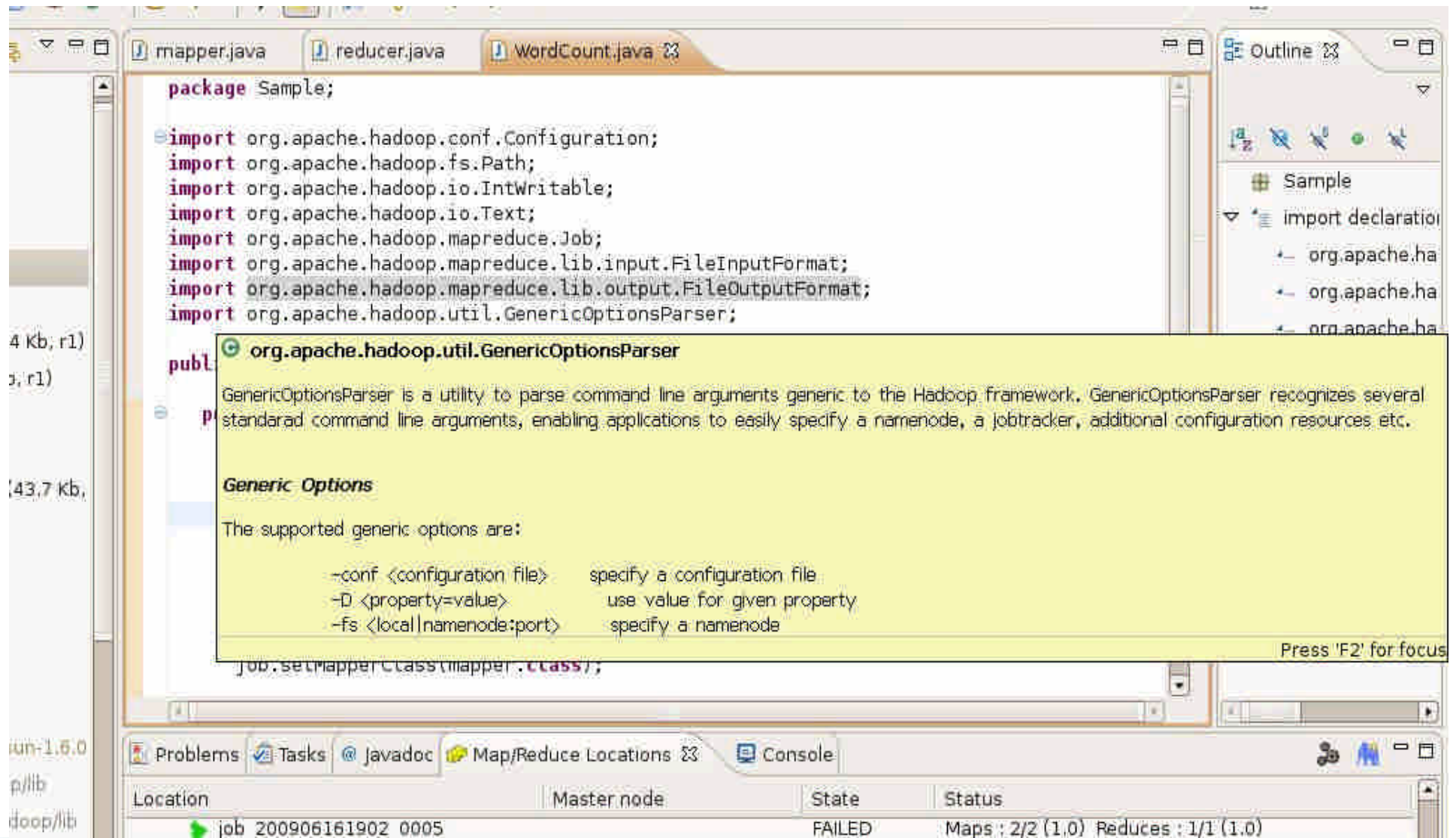
此區為程式窗格

The screenshot shows an IDE interface with a central code editor window titled 'WordCount.java'. The code editor contains the following Java code:

```
//1. 在hdfs 上來源檔案的路徑為 你所指定的 <input>  
//請注意必須先放資料到此hdfs上的資料夾內, 且此資料夾內只能放檔案, 不可再放資料夾  
//2. 運算完後, 程式將執行結果放在hdfs 的輸出路徑為 你所指定的 <output>  
//  
public class WordCount {  
  
    public static class TokenizerMapper extends  
        Mapper<Object, Text, Text, IntWritable> {  
  
        private final static IntWritable one = new IntWritable(1);  
        private Text word = new Text();  
  
        public void map(Object key, Text value, Context context)  
            throws IOException, InterruptedException {  
            StringTokenizer itr = new StringTokenizer(value.toString());  
            while (itr.hasMoreTokens()) {  
                word.set(itr.nextToken());  
                context.write(word, one);  
            }  
        }  
    }  
  
    public static class IntSumReducer extends  
        Reducer<Text, IntWritable, Text, IntWritable> {  
        private IntWritable result = new IntWritable();  
  
        public void reduce(Text key, Iterable<IntWritable> values,  
            Context context) throws IOException, InterruptedException {  
            int sum = 0;  
            for (IntWritable val : values) {  
                sum += val.get();  
            }  
            result.set(sum);  
            context.write(key, result);  
        }  
    }  
}
```

The code editor is surrounded by other IDE components: a Project Explorer on the left showing a file tree with 'WordCount.java' selected, an Outline on the right showing the class structure, and a bottom status bar with tabs for 'Problems', 'Tasks', 'Javadoc', 'Console', and 'Map/Reduce Locations'. The status bar also displays 'Location', 'Master node', 'State', and 'Status' information.

7.2 補充：若之前doc部份設定正確，則滑鼠移至程式碼可取得API完整說明



The screenshot shows an IDE window with three tabs: `mapper.java`, `reducer.java`, and `WordCount.java`. The `WordCount.java` tab is active, displaying the following code:

```
package Sample;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount {
    // ...
    job.setMapperClass(mapper.class);
    // ...
}
```

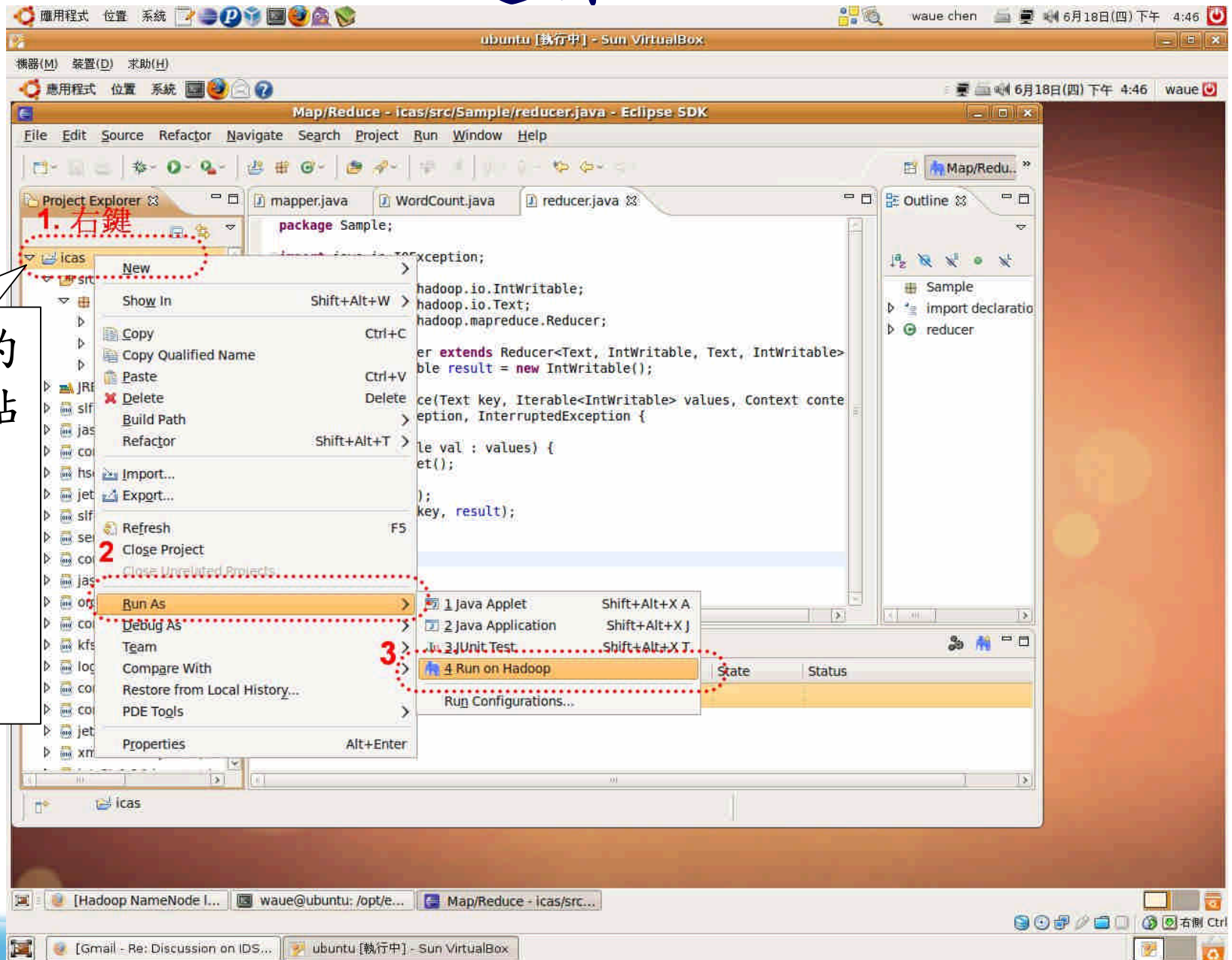
A tooltip is displayed over the `GenericOptionsParser` import, showing the following information:

- org.apache.hadoop.util.GenericOptionsParser**
- GenericOptionsParser is a utility to parse command line arguments generic to the Hadoop framework. GenericOptionsParser recognizes several standard command line arguments, enabling applications to easily specify a namenode, a jobtracker, additional configuration resources etc.
- Generic Options**
- The supported generic options are:
 - `-conf <configuration file>` specify a configuration file
 - `-D <property=value>` use value for given property
 - `-fs <local|namenode:port>` specify a namenode

At the bottom of the IDE, there is a **Map/Reduce Locations** tab showing a table of job details:

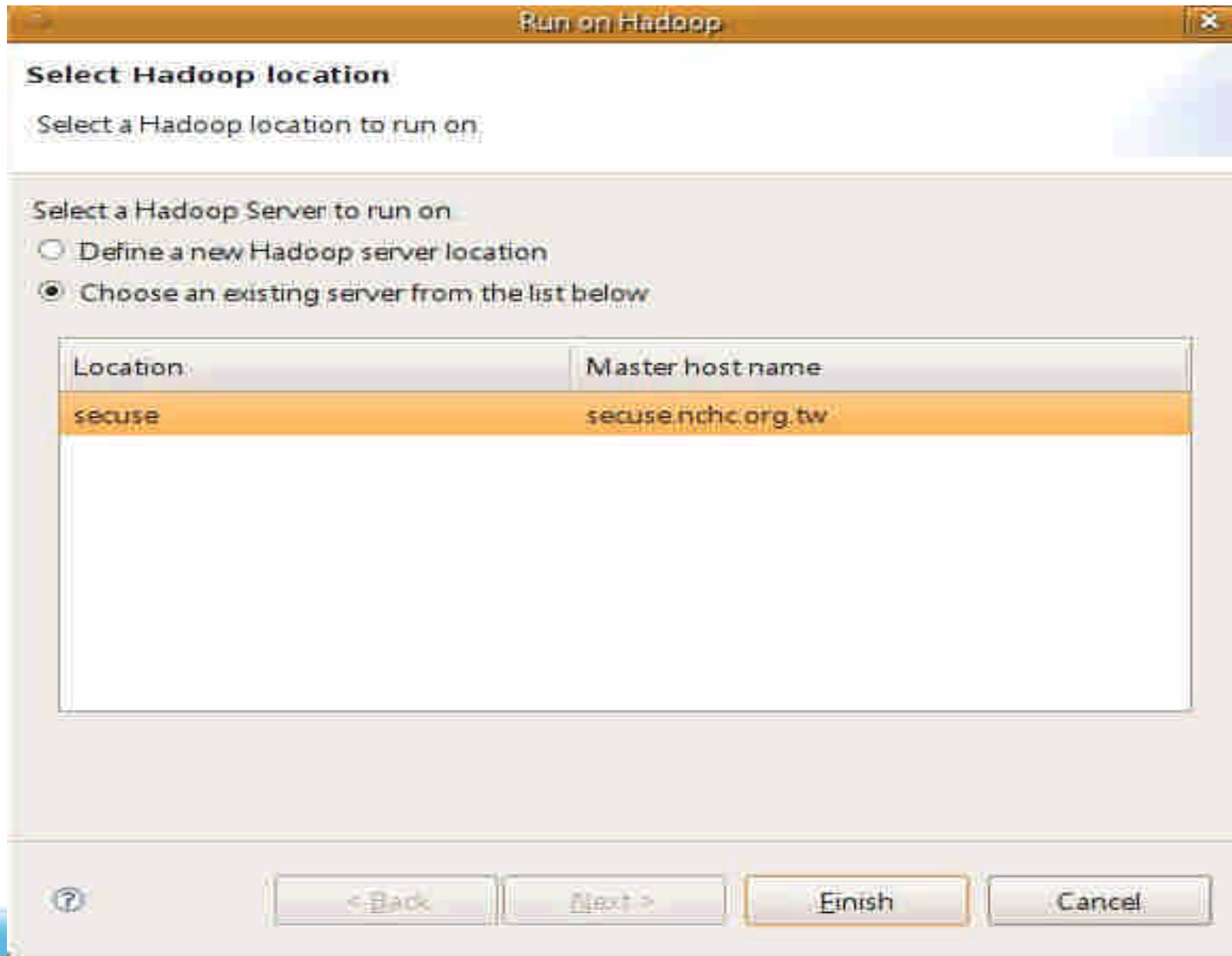
Location	Master node	State	Status
job 200906161902 0005		FAILED	Maps : 2/2 (1.0) Reduces : 1/1 (1.0)

8. 運作

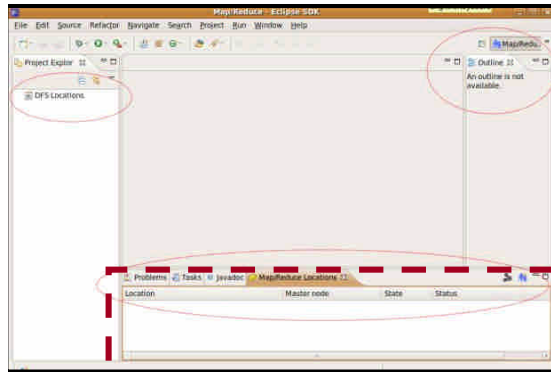


於欲運算的
程式碼處點
選右鍵 →
Run As →
Run on
Hadoop

8-1 選擇之前設定好所要運算的主機



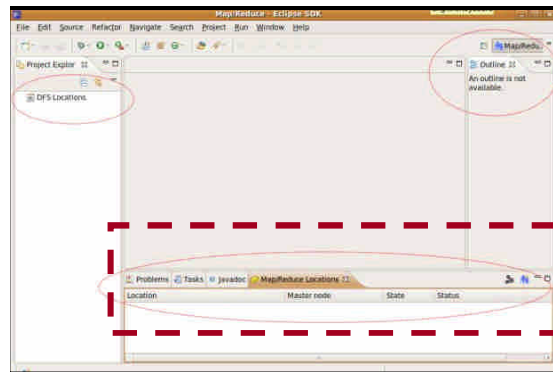
8.2 運算資訊出現於Eclipse 右下方 的Console 視窗



放大

```
Problems Tasks Javadoc Console Map/Reduce Locations
<terminated> WordCount (1) [Java Application] /usr/lib/jvm/java-6-sun-1.6.0.16/bin/java (2010/1/20 下午 6:15:07)
10/01/20 18:15:08 WARN conf.Configuration: DEPRECATED: hadoop-site.xml found in the classpath. Usage of hadoop
10/01/20 18:15:08 WARN mapred.JobConf: The variable mapred.task.maxvmem is no longer used. Instead use mapred
10/01/20 18:15:08 WARN mapred.JobConf: The variable mapred.task.maxvmem is no longer used. Instead use mapred
10/01/20 18:15:08 INFO input.FileInputFormat: Total input paths to process : 2
10/01/20 18:15:08 INFO mapred.JobClient: Running job: job_201001181452_0078
10/01/20 18:15:09 INFO mapred.JobClient: map 0% reduce 0%
10/01/20 18:15:16 INFO mapred.JobClient: map 100% reduce 0%
10/01/20 18:15:28 INFO mapred.JobClient: map 100% reduce 100%
10/01/20 18:15:30 INFO mapred.JobClient: Job complete: job_201001181452_0078
10/01/20 18:15:30 INFO mapred.JobClient: Counters: 17
10/01/20 18:15:30 INFO mapred.JobClient: Job Counters
10/01/20 18:15:30 INFO mapred.JobClient: Launched reduce tasks=1
10/01/20 18:15:30 INFO mapred.JobClient: Launched map tasks=2
10/01/20 18:15:30 INFO mapred.JobClient: Data-local map tasks=2
10/01/20 18:15:30 INFO mapred.JobClient: FileSystemCounters
10/01/20 18:15:30 INFO mapred.JobClient: FILE_BYTES_READ=153
```


8.3 剛剛運算的結果出現如下圖



```
String[] otherArgs = new GenericOptionsParser(
    .getRemainingArgs());
if (otherArgs.length != 0) {
```

Location	Master node	State	Status
secuse	secuse.nchc.org.tw		
job_201001181452_0069		SUCCEEDED	Maps: 2/2 (1.0) Reduces: 1/1 (1.0)

Conclusions

● 優點

- ◆ 快速開發程式
- ◆ 易於除錯
- ◆ 智慧尋找函式庫
- ◆ 自動鍊結API
- ◆ 直接操控 HDFS 與 JobTracker
- ◆ ...

● 缺點

- ◆ Plugin 並會因Eclipse 版本而有不同的狀況

Map Reduce

程式架構

王耀聰 陳威宇

Jazz@nchc.org.tw

waue@nchc.org.tw

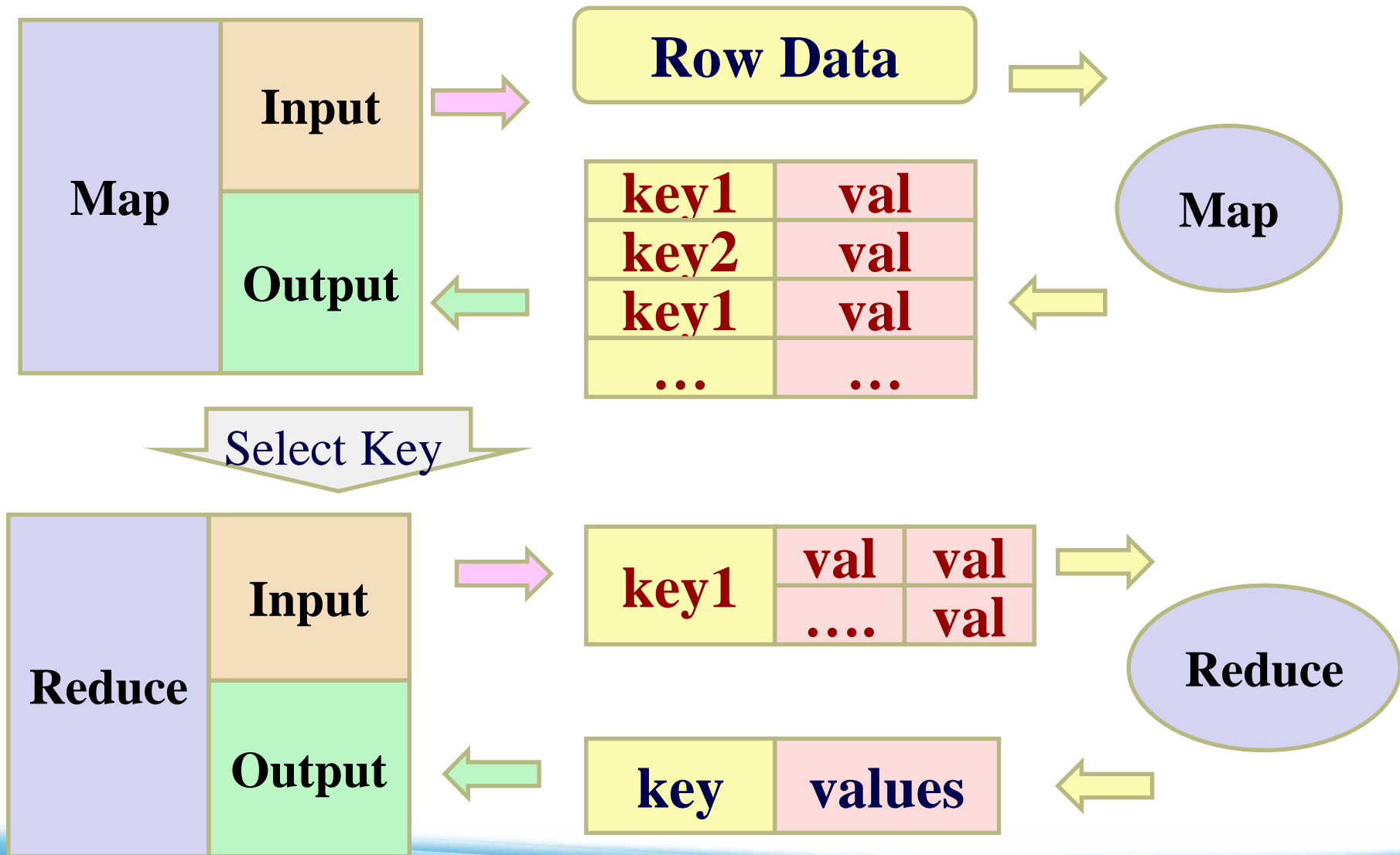


財團法人國家實驗研究院

國家高速網路與計算中心

NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING

<Key, Value> Pair



Program Prototype (v 0.20)

```
Class MR{  
    static public Class Mapper ...{  
    }  
    static public Class Reducer ...{  
    }  
    main()  
        Configuration conf = new Configuration();  
        Job job = new Job(conf, "job name");  
        job.setJarByClass(thisMainClass.class);  
        job.setMapperClass(Mapper.class);  
        job.setReducerClass(Reducer.class);  
        FileInputFormat.addInputPaths(job, new Path(args[0]));  
        FileOutputFormat.setOutputPath(job, new Path(args[1]));  
        其他的設定參數程式碼  
        job.waitForCompletion(true);  
    }  
}
```

Map 區

Reduce 區

設定區

Map 程式碼

Reduce 程式碼

Class Mapper (v 0.20)

```
import org.apache.hadoop.mapreduce.Mapper;
```

```
1 class MyMap extends  
   Mapper < INPUT KEY Class, INPUT VALUE Class, OUTPUT KEY Class, OUTPUT VALUE Class >  
2 {  
3   // 全域變數區  
4   public void map ( INPUT KEY Class key, INPUT VALUE Class value,  
   Context context ) throws IOException, InterruptedException  
5   {  
6     // 區域變數與程式邏輯區  
7     context.write( NewKey, NewValue);  
8   }  
9 }
```

Class Reducer (v 0.20)

```
import org.apache.hadoop.mapreduce.Reducer;
```

```
1 class MyRed extends
```

```
    Reducer < INPUT KEY Class, INPUT VALUE Class, OUTPUT KEY Class, OUTPUT VALUE Class >
```

```
2 {
```

```
3 // 全域變數區
```

```
4 public void reduce ( INPUT KEY Class key, Iterable< INPUT VALUE Class > values,  
    Context context) throws IOException, InterruptedException
```

```
{
```

```
5 // 區域變數與程式邏輯區
```

```
6 context.write( NewKey, NewValue);
```

```
7 }
```

```
8 }
```

```
9
```

其他常用的設定參數

- 設定 Combiner

- ◆ `Job.setCombinerClass (...);`

- 設定 output class

- ◆ `Job.setMapOutputKeyClass(...);`

- ◆ `Job.setMapOutputValueClass(...);`

- ◆ `Job.setOutputKeyClass(...);`

- ◆ `Job.setOutputValueClass(...);`

Class Combiner

- 指定一個combiner，它負責對中間過程的輸出進行聚集，這會有助於降低從Mapper到Reducer數據傳輸量。
- 可不用設定交由Hadoop預設
- 也可不實做此程式，引用Reducer
- 設定
 - ◆ `JobConf.setCombinerClass(Class)`

範例一 (1) - mapper

```
public class HelloHadoop {  
  
    static public class HelloMapper extends  
        Mapper<LongWritable, Text, LongWritable, Text> {  
        public void map(LongWritable key, Text value, Context context)  
            throws IOException, InterruptedException {  
            context.write((LongWritable) key, (Text) value);  
        }  
    } // HelloReducer end  
}
```

..(待續) ...

範例一 (2) - reducer

```
static public class HelloReducer extends
    Reducer<LongWritable, Text, LongWritable, Text> {
public void reduce(LongWritable key, Iterable<Text> values,
    Context context) throws IOException, InterruptedException {
    Text val = new Text();
    for (Text str : values) {
        val.set(str.toString());
    }
    context.write(key, val);
}
} // HelloReducer end
```

..(待續) ...

範例一 (3) - main

```
public static void main(String[] args) throws IOException,  
    InterruptedException, ClassNotFoundException {  
    Configuration conf = new Configuration();  
    Job job = new Job(conf, "Hadoop Hello World");  
    job.setJarByClass(HelloHadoop.class);  
    FileInputFormat.setInputPaths(job, "input");  
    FileOutputFormat.setOutputPath(job, new Path("output-hh1"));  
    job.setMapperClass(HelloMapper.class);  
    job.setReducerClass(HelloReducer.class);  
    job.waitForCompletion(true);  
    } // main end  
} // wordcount class end  
// 完
```

程式設計一

HDFS 操作篇

© TemplatesW.com



財團法人國家實驗研究院

國家高速網路與計算中心

NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING

傳送檔案至HDFS

// 將檔案從local上傳到 hdfs , src 為 local 的來源
， dst 為 hdfs 的目的端

```
public class PutToHdfs {  
    static boolean putToHdfs(String src, String dst, Configuration conf) {  
        Path dstPath = new Path(dst);  
        try {  
            // 產生操作hdfs的物件  
            FileSystem hdfs = dstPath.getFileSystem(conf);  
            // 上傳  
            hdfs.copyFromLocalFile(false, new Path(src), new Path(dst));  
        } catch (IOException e) {  
            e.printStackTrace();  
            return false;  
        }  
        return true;  
    }  
}
```

從HDFS取回檔案

// 將檔案從hdfs下載回local, src 為 hdfs的來源,
dst 為 local 的目的端

```
public class GetFromHdfs {  
    static boolean getFromHdfs(String src,String dst, Configuration conf) {  
        Path dstPath = new Path(src);  
        try {  
            // 產生操作hdfs的物件  
            FileSystem hdfs = dstPath.getFileSystem(conf);  
            // 下載  
            hdfs.copyToLocalFile(false, new Path(src),new Path(dst));  
        } catch (IOException e) {  
            e.printStackTrace();  
            return false;  
        }  
        return true;  
    }  
}
```

檢查與刪除檔案

// checkAndDelete函式，檢查是否存在該資料夾，若有則刪除之

```
public class CheckAndDelete {
    static boolean checkAndDelete(final String path, Configuration conf) {
        Path dst_path = new Path(path);
        try {
            // 產生操作hdfs的物件
            FileSystem hdfs = dst_path.getFileSystem(conf);
            // 檢查是否存在
            if (hdfs.exists(dst_path)) {
                // 有則刪除
                hdfs.delete(dst_path, true);
            } } catch (IOException e) {
                e.printStackTrace();
            return false;
        } return true; }
}
```


程式設計二

範例程式



財團法人國家實驗研究院

國家高速網路與計算中心

NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING

範例二 (1) HelloHadoopV2

說明：

此程式碼比HelloHadoop 增加了

- * 檢查輸出資料夾是否存在並刪除
- * input 資料夾內的資料若大於兩個，則資料不會被覆蓋
- * map 與 reduce 拆開以利程式再利用

測試方法：

將此程式運作在hadoop 0.20 平台上，執行：

```
-----  
hadoop jar V2.jar HelloHadoopV2  
-----
```

注意：

1. 在hdfs 上來源檔案的路徑為 "/user/\$YOUR_NAME/input"，請注意必須先放資料到此hdfs上的資料夾內，且此資料夾內只能放檔案，不可再放資料夾
2. 運算完後，程式將執行結果放在hdfs 的輸出路徑為 "/user/\$YOUR_NAME/output-hh2"

範例二 (2)

```
public class HelloHadoopV2 {
public static void main(String[] args) throws
    IOException,
    InterruptedException, ClassNotFoundException
    {
    Configuration conf = new Configuration();
    Job job = new Job(conf, "Hadoop Hello World
    2");
    job.setJarByClass(HelloHadoopV2.class);
    // 設定 map and reduce 以及 Combiner class
    job.setMapperClass(HelloMapperV2.class);
    job.setCombinerClass(HelloReducerV2.class);
    job.setReducerClass(HelloReducerV2.class);
    // 設定map的輸出型態
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(Text.class);
    // 設定reduce的輸出型態
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);
```

```
FileInputFormat.addInputPath
    (job, new Path("input"));
FileOutputFormat.setOutputPath
    (job, new Path("output-hh2"));
// 呼叫checkAndDelete函式，
// 檢查是否存在該資料夾，若有則刪除之
CheckAndDelete.checkAndDelete("output-hh2",
    conf);
boolean status = job.waitForCompletion(true);
if (status) {
    System.err.println("Integrate Alert Job Finished
    !");
} else {
    System.err.println("Integrate Alert Job Failed
    !");
    System.exit(1);
} } }
```

範例二 (3)

```
public class HelloMapperV2 extends
    Mapper <LongWritable, Text, Text,
    Text> {
    public void map(LongWritable key, Text
    value, Context context)
    throws IOException,
    InterruptedException {
    context.write(new Text(key.toString()),
    value);
    } }
}
```

```
public class HelloReducerV2 extends
    Reducer<Text, Text, Text, Text> {
    public void reduce(Text key, Iterable<Text>
    values, Context context)
    throws IOException, InterruptedException {
    String str = new String("");
    Text final_key = new Text();
    Text final_value = new Text();
    // 將key值相同的values，透過 && 符號分隔
    之
    for (Text tmp : values) {
    str += tmp.toString() + " &&";
    }
    final_key.set(key);
    final_value.set(str);
    context.write(final_key, final_value);
    } }
}
```

範例三 (1) HelloHadoopV3

說明：

此程式碼再利用了 HelloHadoopV2 的 map, reduce 檔，並且自動將檔案上傳到hdfs上運算並自動取回結果，還有提示訊息、參數輸入與印出運算時間的功能

測試方法：

將此程式運作在hadoop 0.20 平台上，執行：

```
-----  
hadoop jar V3.jar HelloHadoopV3 <local_input> <local_output>  
-----
```

注意：

1. 第一個輸入的參數是在local的輸入資料夾，請確認此資料夾內有資料並無子目錄
2. 第二個輸入的參數是在local的運算結果資料夾，由程式產生不用事先建立，若有請刪除之

範例三 (2)

```
public class HelloHadoopV3 {
    public static void main(String[] args) throws
        IOException,
        InterruptedException, ClassNotFoundException
    {
        String hdfs_input = "HH3_input";
        String hdfs_output = "HH3_output";
        Configuration conf = new Configuration();
        // 宣告取得參數
        String[] otherArgs = new
            GenericOptionsParser(conf, args)
                .getRemainingArgs();
        // 如果參數數量不為2 則印出提示訊息
        if (otherArgs.length != 2) {
            System.err
                .println("Usage: hadoop jar
                    HelloHadoopV3.jar <local_input>
                    <local_output>");
            System.exit(2);
        }
    }
}
```

```
Job job = new Job(conf, "Hadoop Hello World");
job.setJarByClass(HelloHadoopV3.class);
// 再利用上個範例的map 與 reduce
job.setMapperClass(HelloMapperV2.class);
job.setCombinerClass(HelloReducerV2.class);
job.setReducerClass(HelloReducerV2.class);
// 設定map reduce 的key value輸出型態
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(Text.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(Text.class);
```

範例三 (2)

```
// 用 checkAndDelete 函式防止overhead的錯誤
CheckAndDelete.checkAndDelete(hdfs_input,
    conf);
CheckAndDelete.checkAndDelete(hdfs_output,
    conf);
// 放檔案到hdfs
PutToHdfs.putToHdfs(args[0], hdfs_input, conf);
// 設定hdfs 的輸入輸出來源路定
FileInputFormat.addInputPath(job, new
    Path(hdfs_input));
FileOutputFormat.setOutputPath(job, new
    Path(hdfs_output));
long start = System.nanoTime();
job.waitForCompletion(true);
```

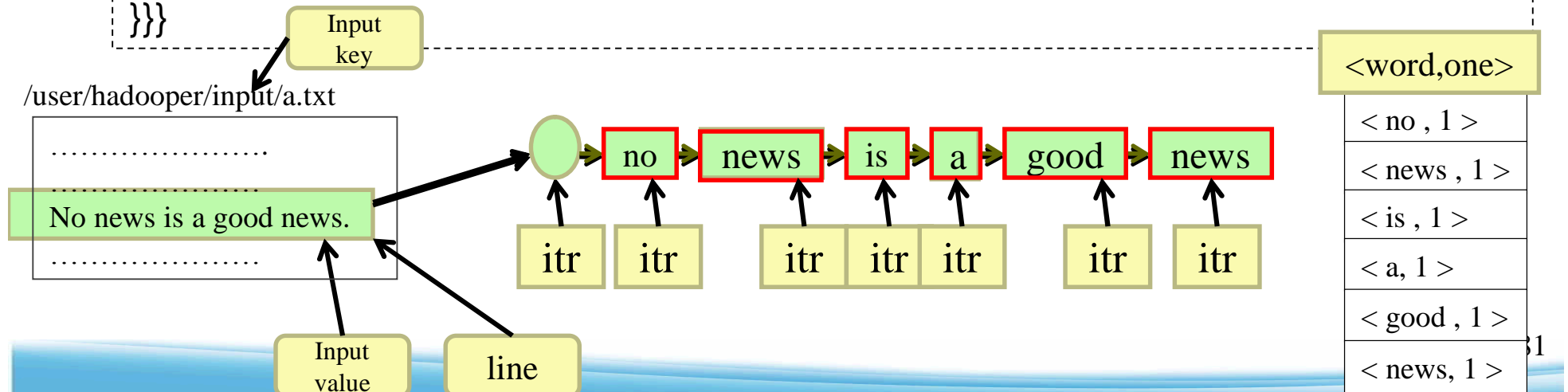
```
// 把hdfs的結果取下
GetFromHdfs.getFromHdfs(hdfs_output, args[1],
    conf);
boolean status = job.waitForCompletion(true);
// 計算時間
if (status) {
    System.err.println("Integrate Alert Job Finished
        !");
    long time = System.nanoTime() - start;
    System.err.println(time * (1E-9) + " secs.");
} else {
    System.err.println("Integrate Alert Job Failed !");
    System.exit(1);
} } }
```

範例四 (1)

```
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
    if (otherArgs.length != 2) {
        System.err.println("Usage: hadoop jar WordCount.jar <input> <output>");
        System.exit(2);
    }
    Job job = new Job(conf, "Word Count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    CheckAndDelete.checkAndDelete(args[1], conf);
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```


範例四 (2)

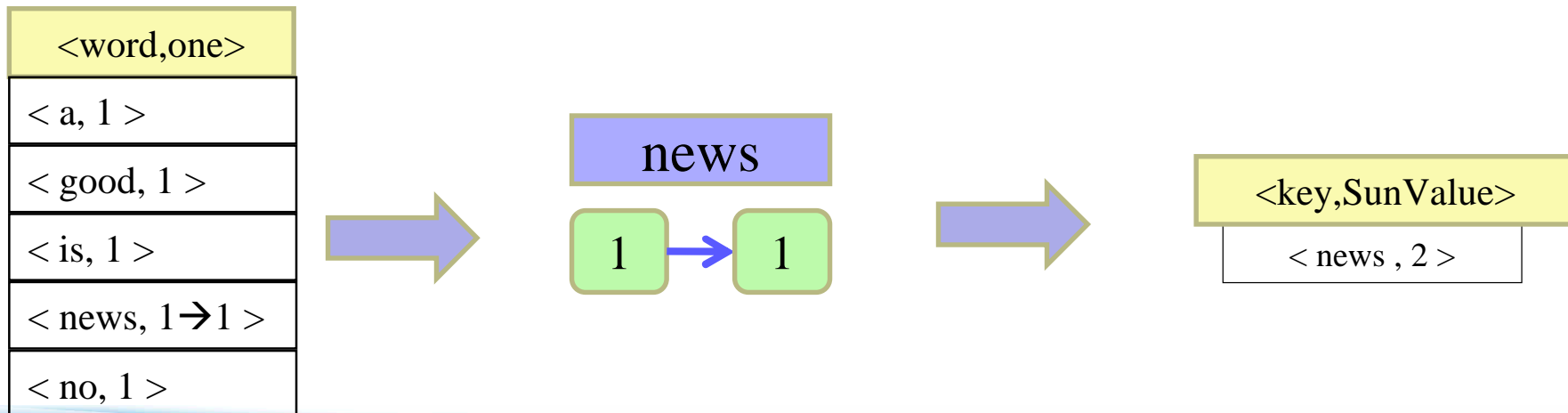
```
1 class TokenizerMapper extends MapReduceBase implements  
  Mapper<LongWritable, Text, Text, IntWritable> {  
2     private final static IntWritable one = new IntWritable(1);  
3     private Text word = new Text();  
4     public void map( LongWritable key, Text value, Context context)  
        throws IOException , InterruptedException {  
5         String line = ((Text) value).toString();  
6         StringTokenizer itr = new StringTokenizer(line);  
7         while (itr.hasMoreTokens()) {  
8             word.set(itr.nextToken());  
9             context.write(word, one);  
        }  
    }  
}
```



範例四 (3)

```
1 class IntSumReducer extends Reducer< Text, IntWritable, Text, IntWritable> {  
2     IntWritable result = new IntWritable();  
3     public void reduce( Text key, Iterable <IntWritable> values, Context context)  
4         throws IOException, InterruptedException {  
5         int sum = 0;  
6         for ( IntWritable val : values )  
7             sum += val.get();  
8         result.set(sum);  
9         context.write ( key, result);  
10    }  
11 }
```

The code above shows two versions of the `reduce` method's loop. The original code (lines 5-7) uses a `for` loop with an `IntWritable` iterator. A red arrow points to a modified version (lines 5-7 in the red dashed box) that uses a `for` loop with an integer index `i` to access `values[i].get()`.



範例五 (1) WordCountV2

說明：

用於字數統計，並且增加略過大小寫辨識、符號篩除等功能

測試方法：

將此程式運作在hadoop 0.20 平台上，執行：

```
-----  
hadoop jar WCV2.jar WordCountV2 -Dwordcount.case.sensitive=false \  
<input> <output> -skip patterns/patterns.txt  
-----
```

注意：

1. 在hdfs 上來源檔案的路徑為 你所指定的 <input>
請注意必須先放資料到此hdfs上的資料夾內，且此資料夾內只能放檔案，不可再放資料夾
2. 運算完後，程式將執行結果放在hdfs 的輸出路徑為 你所指定的 <output>
3. 請建立一個資料夾 pattern 並在裡面放置pattern.txt，內容如

(一行一個，前置提示符號\) →



範例五 (2)

```
public class WordCountV2 extends Configured implements Tool {
    public static class Map extends MapReduceBase implements
        Mapper<LongWritable, Text, Text, IntWritable> {
        static enum Counters { INPUT_WORDS }
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();
        private boolean caseSensitive = true;
        private Set<String> patternsToSkip = new HashSet<String>();
        private long numRecords = 0;
        private String inputFile;
        public void configure(JobConf job) {
            caseSensitive = job.getBoolean("wordcount.case.sensitive", true);
            inputFile = job.get("map.input.file");
            if (job.getBoolean("wordcount.skip.patterns", false)) {
                Path[] patternsFiles = new Path[0];
                try {
                    patternsFiles = DistributedCache.getLocalCacheFiles(job);
                } catch (IOException ioe) {
                    System.err
                        .println("Caught exception while getting cached files: "
                            + StringUtils.stringifyException(ioe));
                }
                for (Path patternsFile : patternsFiles) {
                    parseSkipFile(patternsFile);
                }
            }
        }
    }
}
```

```
private void parseSkipFile(Path patternsFile) {
    try {
        BufferedReader fis = new BufferedReader(new FileReader(
            patternsFile.toString()));
        String pattern = null;
        while ((pattern = fis.readLine()) != null) {
            patternsToSkip.add(pattern);
        }
    } catch (IOException ioe) {
        System.err.println("Caught exception while parsing the cached
            file '"+ patternsFile + "' : " + tringUtils.stringifyException(ioe));
    }
}

public void map(LongWritable key, Text value,
    OutputCollector<Text, IntWritable> output, Reporter reporter)
    throws IOException {
    String line = (caseSensitive) ? value.toString() : value.toString()
        .toLowerCase();
    for (String pattern : patternsToSkip)
        line = line.replaceAll(pattern, "");
    StringTokenizer tokenizer = new StringTokenizer(line);
    while (tokenizer.hasMoreTokens()) {
        word.set(tokenizer.nextToken());
        output.collect(word, one);
        reporter.incrCounter(Counters.INPUT_WORDS, 1);
    }
}
```

範例五 (3)

```
if ((++numRecords % 100) == 0) {
    reporter.setStatus("Finished processing " + numRecords
        + " records " + "from the input file: " + inputFile);
} } }
public static class Reduce extends MapReduceBase implements
    Reducer<Text, IntWritable, Text, IntWritable> {
public void reduce(Text key, Iterator<IntWritable> values,
    OutputCollector<Text, IntWritable> output, Reporter reporter)
    throws IOException {
    int sum = 0;
    while (values.hasNext()) {
        sum += values.next().get(); }
    output.collect(key, new IntWritable(sum)); } }
public int run(String[] args) throws Exception {
    JobConf conf = new JobConf(getConf(), WordCount.class);
    conf.setJobName("wordcount");
    String[] otherArgs = new GenericOptionsParser(conf, args)
        .getRemainingArgs();
    if (otherArgs.length < 2) {
        System.out.println("WordCountV2 [-
            Dwordcount.case.sensitive=<false|true> \\ ");
        System.out.println("    <inDir> <outDir> [-skip
            Pattern_file]");
        return 0; }
    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);
```

```
        conf.setMapperClass(Map.class);
        conf.setCombinerClass(Reduce.class);
        conf.setReducerClass(Reduce.class);
        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);
        List<String> other_args = new ArrayList<String>();
        for (int i = 0; i < args.length; ++i) {
            if ("-skip".equals(args[i])) {
                DistributedCache
                    .addCacheFile(new Path(args[++i]).toUri(), conf);
                conf.setBoolean("wordcount.skip.patterns", true);
            } else {other_args.add(args[i]); } }
        FileInputFormat.setInputPaths(conf, new Path(other_args.get(0)));
        FileOutputFormat.setOutputPath(conf, new
            Path(other_args.get(1)));
        CheckAndDelete.checkAndDelete(other_args.get(1), conf);
        JobClient.runJob(conf);
        return 0; }
    public static void main(String[] args) throws Exception {
        int res = ToolRunner.run(new Configuration(), new
            WordCountV2(), args);
        System.exit(res);
    }
}
```

範例六 (1) WordIndex

說明：

將每個字出於哪個檔案，那一行印出來

測試方法：

將此程式運作在hadoop 0.20 平台上，執行：

```
-----  
hadoop jar WI.jar WordIndex <input> <output>  
-----
```

注意：

1. 在hdfs 上來源檔案的路徑為 你所指定的 <input>
請注意必須先放資料到此hdfs上的資料夾內，且此資料夾內只能放檔案，
不可再放資料夾
2. 運算完後，程式將執行結果放在hdfs 的輸出路徑為 你所指定的
<output>

範例六 (2)

```
public class WordIndex {
    public static class wordindexM extends
        Mapper<LongWritable, Text, Text, Text> {
    public void map(LongWritable key, Text value,
        Context context)
        throws IOException, InterruptedException {

        FileSplit fileSplit = (FileSplit)
            context.getInputSplit();

        Text map_key = new Text();
        Text map_value = new Text();
        String line = value.toString();
        StringTokenizer st = new
            StringTokenizer(line.toLowerCase());
        while (st.hasMoreTokens()) {
            String word = st.nextToken();
            map_key.set(word);
            map_value.set(fileSplit.getPath().getName() +
                ":" + line);
            context.write(map_key, map_value);
        } } }
```

```
static public class wordindexR extends
    Reducer<Text, Text, Text, Text> {
    public void reduce(Text key, Iterable<Text>
        values,
        OutputCollector<Text, Text> output, Reporter
        reporter)
        throws IOException {
        String v = "";
        StringBuilder ret = new StringBuilder("\n");
        for (Text val : values) {
            v += val.toString().trim();
            if (v.length() > 0)
                ret.append(v + "\n");
        }
        output.collect((Text) key, new
            Text(ret.toString()));
    } }
```

範例六 (2)

```
public static void main(String[] args) throws
    IOException,
    InterruptedException,
    ClassNotFoundException {
    Configuration conf = new Configuration();
    String[] otherArgs = new
        GenericOptionsParser(conf, args)
            .getRemainingArgs();
    if (otherArgs.length < 2) {
        System.out.println("hadoop jar
            WordIndex.jar <inDir> <outDir>");
        return;
    }
    Job job = new Job(conf, "word index");
    job.setJobName("word inverted index");
    job.setJarByClass(WordIndex.class);
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(Text.class);
```

```
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);
    job.setMapperClass(wordindexM.class);
    job.setReducerClass(wordindexR.class);
    job.setCombinerClass(wordindexR.class);
    FileInputFormat.setInputPaths(job, args[0]);
    CheckAndDelete.checkAndDelete(args[1],
        conf);
    FileOutputFormat.setOutputPath(job, new
        Path(args[1]));
    long start = System.nanoTime();
    job.waitForCompletion(true);
    long time = System.nanoTime() - start;
    System.err.println(time * (1E-9) + " secs.");
}}
```


範例七 (1) TsmMenu

說明：

將之前的功能整合起來

測試方法：

將此程式運作在hadoop 0.20 平台上，執行：

```
hadoop jar TsmMenu.jar <功能>
```

注意：

1. 此程式需與之前的所有範例一起打包成一個jar檔

範例七 (2)

```
public class TsmMenu {
    public static void main(String argv[]) {
        int exitCode = -1;
        ProgramDriver pgd = new ProgramDriver();
        if (argv.length < 1) {

            System.out.print("*****\n");
            + "歡迎使用 TSM 的運算功能\n" + "指令：\n"
            + " Hadoop jar TSM-example-*.jar <功能> \n" + "功能：\n"
            + " HelloHadoop: 秀出Hadoop的<Key,Value>為何\n"
            + " HelloHadoopV2: 秀出Hadoop的<Key,Value> 進階版\n"
            + " HelloHadoopV3: 秀出Hadoop的<Key,Value> 進化版\n"
            + " WordCount: 計算輸入資料夾內分別在每個檔案的\n"
            + " WordCountV2: WordCount 進階版\n"
            + " WordIndex: 索引每個字與其所有出現的所在列\n"
            + "*****\n");
        } else {
```

```
try {
    pgd.addClass("HelloHadoop", HelloHadoop.class, "
Hadoop hello world");
    pgd.addClass("HelloHadoopV2",
HelloHadoopV2.class, " Hadoop hello world V2");
    pgd.addClass("HelloHadoopV3",
HelloHadoopV3.class, " Hadoop hello world V3");
    pgd.addClass("WordCount", WordCount.class, "
word count.");
    pgd.addClass("WordCountV2",
WordCountV2.class, " word count V2.");
    pgd.addClass("WordIndex", WordIndex.class,
"invert each word in line");
    pgd.driver(argv);
    // Success
    exitCode = 0;
    System.exit(exitCode);
} catch (Throwable e) {
    e.printStackTrace();
}}}
```

Conclusions

- 以上範例程式碼包含
 - ◆ Hadoop 的key,value 架構
 - ◆ 操作Hdfs 檔案系統
 - ◆ Map Reduce運算方式
- 執行hadoop 運算時，程式檔不用上傳至hadoop 上，但資料需要再HDFS內
- 可運用範例七的程式達成連續運算
- Hadoop 0.20 與 Hadoop 0.18 有些API有些許差異，盡可能完全改寫

Program Prototype (v 0.18)

```
Class MR{  
  Map  
  區  
  ↑↓  
  Reduce  
  區  
  ↑↓  
  設定  
  區  
  ↓  
  Class Mapper ...{  
    Map 程式碼  
  }  
  Class Reducer ...{  
    Reduce 程式碼  
  }  
  main(){  
    JobConf conf = new JobConf( "MR.class" );  
    conf.setMapperClass(Mapper.class);  
    conf.setReduceClass(Reducer.class);  
    FileInputFormat.setInputPaths(conf, new Path(args[0]));  
    FileOutputFormat.setOutputPath(conf, new Path(args[1]));  
    其他的設定參數程式碼  
    JobClient.runJob(conf);  
  }  
}
```

Class Mapper (v0.18)

```
import org.apache.hadoop.mapred.*;

1 class MyMap extends MapReduceBase
  implements Mapper < INPUT KEY , INPUT VALUE , OUTPUT KEY , OUTPUT VALUE >
2 {
3   // 全域變數區
4   public void map ( INPUT KEY key, INPUT VALUE value,
      OutputCollector< OUTPUT KEY , OUTPUT VALUE > output,
      Reporter reporter) throws IOException
5   {
6     // 區域變數與程式邏輯區
7     output.collect( NewKey, NewValue);
8   }
9 }
```

Class Reducer (v0.18)

```
import org.apache.hadoop.mapred.*;
```

```
1 class MyRed extends MapReduceBase  
implements Reducer < INPUT KEY , INPUT VALUE , OUTPUT KEY , OUTPUT VALUE >  
2 {  
3 // 全域變數區  
4 public void reduce ( INPUT KEY key, Iterator< INPUT VALUE > values,  
OutputCollector< OUTPUT KEY , OUTPUT VALUE > output,  
Reporter reporter) throws IOException  
5 {  
6 // 區域變數與程式邏輯區  
7 output.collect( NewKey, NewValue);  
8 }  
9 }
```

News! Google 獲得MapReduce專利

2010/01/20 ~

- Named” system and method for efficient large-scale data processing”
 - ◆ Patent #7,650,331
- Claimed since 2004
- Not programming language, but the merge methods
- **What does it mean for Hadoop?**

Map Reduce

專案分享

王耀聰 陳威宇

Jazz@nchc.org.tw

waue@nchc.org.tw



財團法人國家實驗研究院

國家高速網路與計算中心

NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING

用Hadoop 打造 Location Plus

2009/11/17

- 「Location Plus！」服務，擊敗了其他38組作品獲得Yahoo開發競賽的優勝
- 從大量的批踢踢BBS文章中，找出臺灣17個城市的熱門話題
- 每個城市提供30個熱門話題和相關的參考詞
- 可以讓使用者用這些熱門話題來搜尋 Yahoo知識+、生活+、無名小站等內容
- 提供了手機版介面，讓使用者到任何地方就知道當地有哪些熱門話題

案例：警訊整合系統

- 目的：
 - ◆ 將原本複雜難懂的警訊日誌整合成易於明瞭的報告
 - ◆ 透過“雲端”來運算大量資料
- 環境：
 - ◆ hadoop 0.20
 - ◆ Java 1.6
 - ◆ Apache 2

輸入資料

[**] [1:538:15] NETBIOS SMB IPC\$ unicode share access [**]
[Classification: Generic Protocol Command Decode] [Priority: 3]
09/04-17:53:56.363811 168.150.177.165:1051 ->
168.150.177.166:139
TCP TTL:128 TOS:0x0 ID:4000 IpLen:20 DgmLen:138 DF
AP Seq: 0x2E589B8 Ack: 0x642D47F9 Win: 0x4241
TcpLen: 20

[**] [1:1917:6] SCAN UPnP service discover attempt [**]
[Classification: Detection of a Network Scan] [Priority: 3]
09/04-17:53:56.385573 168.150.177.164:1032 ->
239.255.255.250:1900
UDP TTL:1 TOS:0x0 ID:80 IpLen:20 DgmLen:161
Len: 133

[**] [1:1917:6] SCAN UPnP service discover attempt [**]
[Classification: Detection of a Network Scan] [Priority: 3]
09/04-17:53:56.386910 168.150.177.164:1032 ->
239.255.255.250:1900
UDP TTL:1 TOS:0x0 ID:82 IpLen:20 DgmLen:161
Len: 133

[**] [1:1917:6] SCAN UPnP service discover attempt [**]
[Classification: Detection of a Network Scan] [Priority: 3]
09/04-17:53:56.388244 168.150.177.164:1032 ->
239.255.255.250:1900
UDP TTL:1 TOS:0x0 ID:84 IpLen:20 DgmLen:161
Len: 133

[**] [1:1917:6] SCAN UPnP service discover attempt [**]
[Classification: Detection of a Network Scan] [Priority: 3]
09/04-17:53:56.417045 168.150.177.164:45461 -> 168.150.177.1:1900
UDP TTL:1 TOS:0x0 ID:105 IpLen:20 DgmLen:161
Len: 133

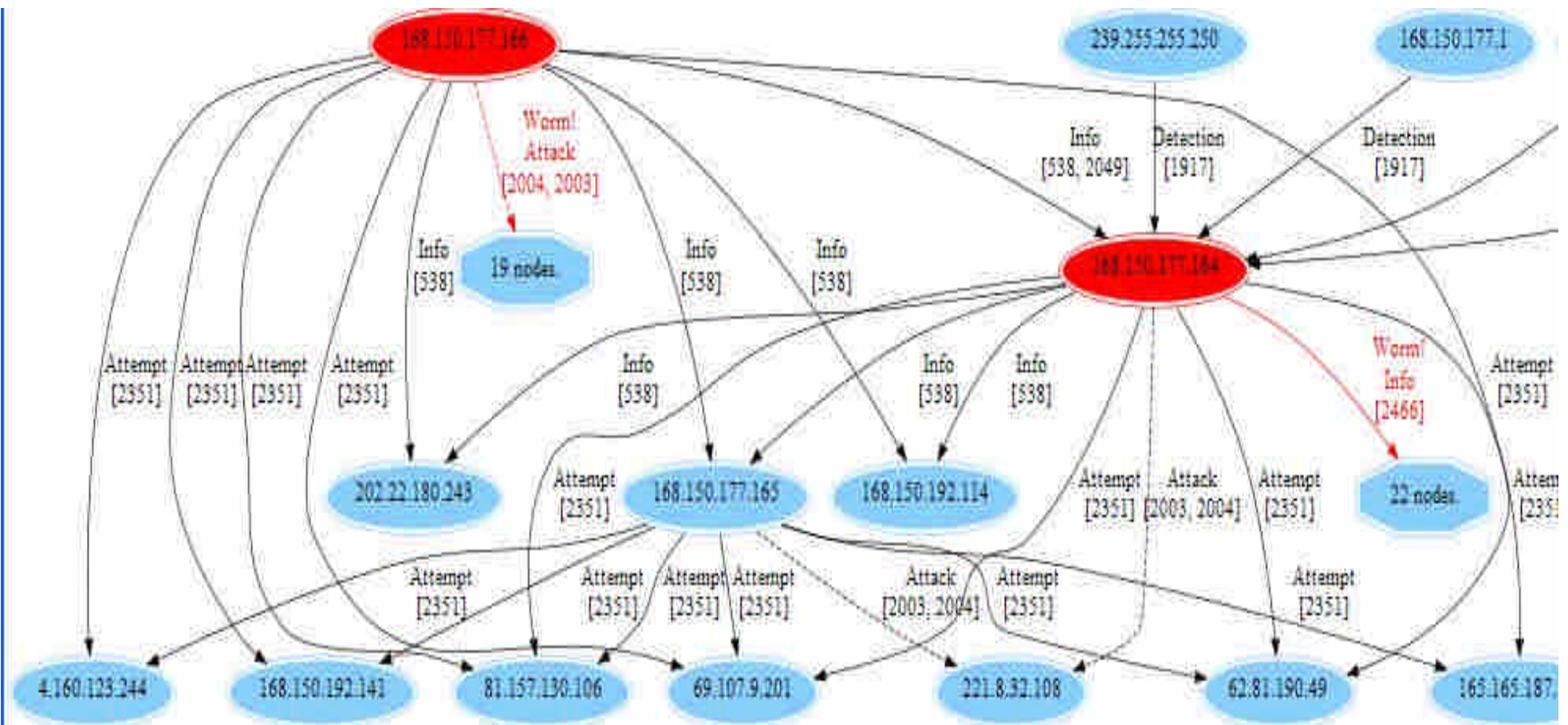
[**] [1:1917:6] SCAN UPnP service discover attempt [**]
[Classification: Detection of a Network Scan] [Priority: 3]
09/04-17:53:56.420759 168.150.177.164:45461 -> 168.150.177.1:1900
UDP TTL:1 TOS:0x0 ID:117 IpLen:20 DgmLen:160
Len: 132

[**] [1:1917:6] SCAN UPnP service discover attempt [**]
[Classification: Detection of a Network Scan] [Priority: 3]
09/04-17:53:56.422095 168.150.177.164:45461 -> 168.150.177.1:1900
UDP TTL:1 TOS:0x0 ID:118 IpLen:20 DgmLen:161
Len: 133

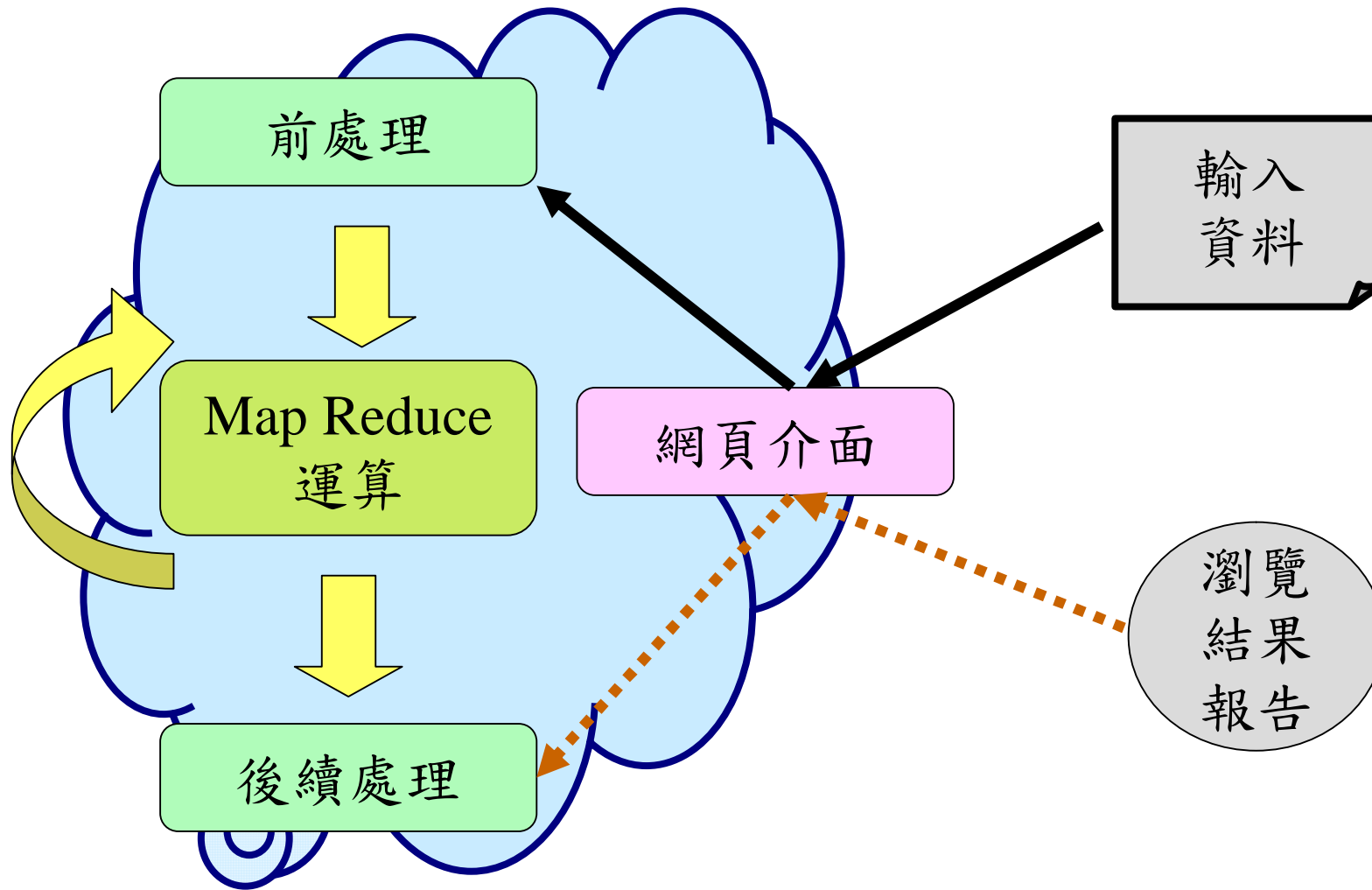
[**] [1:2351:10] NETBIOS DCERPC ISystemActivator path overflow
attempt little endian unicode [**]
[Classification: Attempted Administrator Privilege Gain] [Priority: 1]
09/04-17:53:56.442445 198.8.16.1:10179 -> 168.150.177.164:135
TCP TTL:105 TOS:0x0 ID:49809 IpLen:20 DgmLen:1420 DF
A* Seq: 0xF9589BBF Ack: 0x82CCF5B7 Win: 0xFFFF
TcpLen: 20
[Xref => <http://www.microsoft.com/technet/security/bulletin/MS03-026.msp>][Xref =>
<http://cgi.nessus.org/plugins/dump.php3?id=11808>][Xref =>
<http://cve.mitre.org/cgi-bin/cvename.cgi?name=2003-0352>][Xref =>
<http://www.securityfocus.com/bid/8205>]

輸出資料

Generate dot graph format

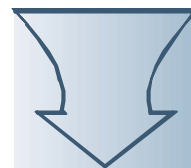


系統分析



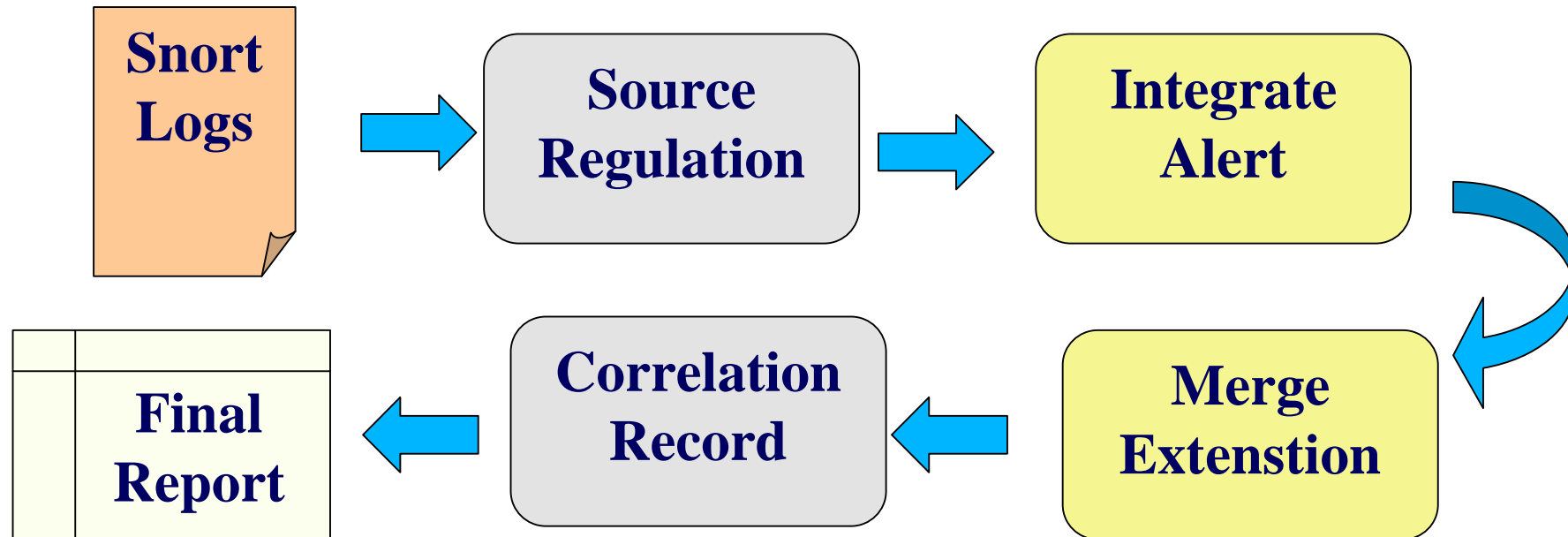
Alert Merge Example

Destination IP	Attack Signature	Source IP	Destination Port	Source Port	Packet Protocol	Timestamp
Host_1	Trojan	Sip1	80	4077	tcp	T1
Host_1	Trojan	Sip2	80	4077	tcp	T2
Host_1	Trojan	Sip1	443	5002	tcp	T3
Host_2	Trojan	Sip1	443	5002	tcp	T4
Host_3	D.D.O.S	Sip3	53	6007	udp	T5
Host_3	D.D.O.S	Sip4	53	6008	tcp	T5
Host_3	D.D.O.S	Sip5	53	6007	udp	T5
Host_3	D.D.O.S	Sip6	53	6008	tcp	T5



Key		Values				
Host_1	Trojan	Sip1,Sip2	80,443	4077,5002	tcp	T1,T2,T3
Host_2	Trojan	Sip1	443	5002	tcp	T4
Host_3	D.D.O.S.	Sip3,Sip4,Sip5 ,Sip6	53	6007,6008	tcp, udp	T5

程式流程圖



結論

- 評估
- 系統分析
 - ◆ 輸入輸出
 - ◆ 系統元件
 - ◆ 各元件參數與串流
- 實做
 - ◆ 多次運算
 - ◆ 前處理與後處理

QUESTIONS
&
THANKS

© TemplatesWan.com



財團法人國家實驗研究院

國家高速網路與計算中心

NATIONAL CENTER FOR HIGH-PERFORMANCE COMPUTING